NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UTILIZING ANDROID AND THE CLOUD COMPUTING ENVIRONMENT TO INCREASE SITUATIONAL AWARENESS FOR A MOBILE DISTRIBUTED RESPONSE

by

Michael Asche
Monique Crewes

March 2012

Thesis Advisor:	Gurminder Singh
Second Reader:	John Gibson

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Utilizing Android and the Cloud Computing Environment to Increase Situational Awareness for a Mobile Distributed Response			**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael Asche, Monique Crewes			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number: N/A.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Maintaining an accurate Common Operational Picture (COP) is a strategic requirement for efficient and successful missions in both disaster response and battlefield scenarios. Past practices include utilizing cellular, radio, and computer based communication methods and updating individual maps accordingly. A drawback of these practices has been interoperability of these devices as well as accurate reporting and documentation among different entities of the effort.

Recent advances in technology have led to the utilization of collaborative maps for maintaining a COP amongst command centers. Despite the advantages this technique offers, it does not address the difficulties surrounding receiving reports from field entities as well as ensuring these entities also have good situational awareness. The goal of this research is to explore smartphone capabilities in conjunction with cloud computing to determine how they can extend the benefits of collaborative maps to mobile users while simultaneously ensuring command centers are receiving accurate, up-to-date reports from the field.

14. SUBJECT TERMS Android Programming, Cloud Computing, Common Operating Picture, Web Programing			**15. NUMBER OF PAGES** 129
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UTILIZING ANDROID AND THE CLOUD COMPUTING ENVIRONMENT TO INCREASE SITUATIONAL AWARENESS FOR A MOBILE DISTRIBUTED RESPONSE

Michael Asche
Lieutenant, United States Navy
B.S., United States Naval Academy, 2006

Monique Crewes
Lieutenant, United States Navy
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2012

Authors: Michael Asche
Monique Crewes

Approved by: Gurminder Singh
Thesis Advisor

John Gibson
Second Reader

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Maintaining an accurate Common Operational Picture (COP) is a strategic requirement for efficient and successful missions in both disaster response and battlefield scenarios. Past practices include utilizing cellular, radio, and computer based communication methods and updating individual maps accordingly. A drawback of these practices has been interoperability of these devices as well as accurate reporting and documentation among different entities of the effort.

Recent advances in technology have led to the utilization of collaborative maps for maintaining a COP amongst command centers. Despite the advantages this technique offers, it does not address the difficulties surrounding receiving reports from field entities as well as ensuring these entities also have good situational awareness. The goal of this research is to explore smartphone capabilities in conjunction with cloud computing to determine how they can extend the benefits of collaborative maps to mobile users while simultaneously ensuring command centers are receiving accurate, up-to-date reports from the field.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

Advanced Ground Information Systems (AGIS)

Application Programming Interface (API)

Blue Force Tracking (BFT)

California Department of Forestry and Fire Protection (CAL FIRE)

Commercial-Off-The-Shelf (COTS)

Global Command and Control System (GCCS)

Global Positioning System (GPS)

Google App Engine (GAE)

Google Web Toolkit (GWT)

Infrastructure as a Service (IaaS)

Next Generation Incident Command System (NICS)

Not-so-Relational (NSR)

Operating System (OS)

Platform as a Service (PaaS)

Point of Interest (POI)

Smartphone Assisted Readiness, Command, and Control System (SPARCCS)

Software Development Kit (SDK)

Structured Query Language (SQL)

User Interface Designer (UID)

Worldwide Military Command and Control System (WMCCS)

XMLHttpRequests (XHR)

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Dr. Gurminder Singh and Professor John H. Gibson for their guidance and support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

1. Situational Awareness

a. Definition

Over the past several decades the use of the term situational awareness has been expanded to apply to crisis situations both military and civilian. A widely used definition of this term developed by Endsley, a leading researcher at the U.S. Army Research Institute, is

> the perception of the elements in the environment within a volume of space and time, the comprehension of their meaning, and the projection of their status in the near future. (Endsley 1995, 36)

Following this definition, Endsley expanded her definition into a model of situational awareness consisting of three levels. The first level is the perception of the elements of the current situation. This is an understanding of the physical environment in which the situation is occurring. The second level is an understanding of the situation. Here the dynamics of the physical elements and the people involved in the situation must be comprehended, in terms of both their movement and purpose. The third level is the projection of future status of the situation. Since situational awareness develops over time, the effect of current events has an impact on what occurs in the near future.

b. Importance

Maintaining situational awareness is critical to decision makers as it allows them to understand the current operating environment and make critical, real-time decisions based on what is occurring at the present time and what could occur in the future. In addition it gives them the ability to get immediate feedback on past decisions enabling them to make better choices in the future.

2. Common Operating Picture

Situational awareness is enhanced through the use of a common operating picture. Usually when a disaster, emergency, and/or military response is needed, multiple agencies, often times both local and national, must keep track of what is happening and where it is happening. The cooperating agencies must come together to get a shared situational awareness, or what is known as a common operating picture. With this common operating picture it becomes easier for agencies to communicate and resolve the issues they are facing more effectively, saving both lives and resources in the process.

a. Difficulties

However, gaining an accurate common operating picture can be very difficult. This is because with the introduction of multiple agencies into a crisis situation interoperability becomes the single overwhelming issue preventing interagency cooperation. The Chairman of the Joint Chiefs of Staff Instruction 3152.01A, which pertains

to the Global Command and Control System Common Operating Picture Reporting Requirements, defines common operating picture as

> a distributed data processing and exchange environment for developing a dynamic database of objects, allowing each user to filter and contribute to this database, according to the user's area of responsibility and command role. The common operational picture provides the integrated capability to receive, correlate, and display a common tactical picture, including planning applications and theater-generated overlays and projections. (Department of Defense, 2008)

Using this definition each agency has developed its own respective methodology and corresponding systems for achieving a common operating picture. These corresponding systems and databases are often unable to communicate with each other. As a consequence, information dissemination across agencies becomes almost impossible in real time, leaving decision makers in a precarious position of having to make decisions with untimely, and possibly inaccurate information. This can be seen in efforts during the response to bombings in New York on September 11, 2001, when police, fire and emergency personnel struggled to communicate with one another regarding where to go and which agency was in charge of doing what aspect of relief. This was not only relevant across the respective responder communities, but also within individual responder communities. For example, fire departments from different counties had trouble communicating using the same radios due to the inability to construct a cohesive communications plan. On a larger scale, during Hurricane Katrina this became even more evident as local and state disaster

response teams were completely overwhelmed and national level response was needed. It was difficult for a common operating picture to be constructed between all three tiers of governmental responders (Copeland 2008, 1).

b. Solutions

In an effort to relieve interagency interoperability, a system should be adopted using commercial-off-the-shelf (COTS) smartphones in conjunction with a cloud-computing server. With the recent upgrades in smartphone technology, many pieces of data needed to craft a common operating picture can be gathered using this widespread, relatively cheap technology. Using the smartphones' global positioning features, locations of points of interest relevant to decision can be gathered. Likewise, pictures and video can be passed with ease and without the need to carry multiple devices. Furthermore, by using a common communication technology there is no need to develop a complicated communications plan and generate an ad-hoc communications infrastructure that spans across all agencies.

Using smartphones in conjunction with a cloud computing platform can further aid in the creation of an effective common operating picture and the improvement of overall situational awareness. By utilizing a web-based cloud computing infrastructure, information can be made easily available, as the only thing needed to look at and update information would be access to the Internet: not a particular computer, place or command post. Another benefit of cloud computing is reduced costs, as it pools computing resources that can be distributed to applications as

needed. Deployment of the application will be faster, as there is no need to update all networking hardware and software within a given network infrastructure. Likewise the technology is easily scalable, so if demand for information increases then the infrastructure will be able to handle the demand without needing to update the architecture of the system. Most importantly, it has redundancy so that, in the case the crisis arises in the area of the cloud service provider, disaster recovery services can bring up a separate server immediately.

B. FOCUS

In this thesis we will develop and demonstrate the feasibility of a cloud-based system, known herein as the Smartphone Assisted Readiness, Command, and Control System (SPARCCS), to advance the SA capability of rapid response personnel. This system will allow for Android smartphones and a central web-based cloud application to communicate. We will use our web-based cloud application as the command center to effectively display information and communicate with Android smartphones to create a common operating picture thereby improving situational awareness.

2. Organization

a. Chapter II (Evolution of Command and Control Systems)

The second chapter of this thesis will highlight past and existing Command and Control technologies. It will touch on their strengths and weakness, further

demonstrating a need for the SPARCCS program. It will additionally put into context related work within the SA development community.

b. Chapter III (Infrastructure Background)

The third chapter of this thesis will describe cloud computing and its database storage component. It will further delve into the Google App Engine, the cloud platform by which the web-based application is supported. It will likewise describe the Android operating system used by the smartphones in the SPARCCS infrastructure. The chapter will conclude with details about the current state of web services used by the SPARCCS environment.

c. Chapter IV (SPARCCS Implementation)

The fourth chapter of this thesis will describe the application proposed and demonstrated by this thesis. It will define the software methodology behind the Android mobile application and the supporting web-based cloud application. It will also outline the server communication structure that these applications use to communicate.

d. Chapter V (Conclusions and Future Work)

The fifth chapter of this thesis will begin with an overall summary of the SPARCCS system. It will highlight strengths and weakness of the infrastructure and provide guidance for future work within the SPARCCS framework. The thesis will then conclude with an overarching look at the current state of Common Operating Picture programs and will detail how the SPARCCS program attempts to alleviate some of the pitfalls in the existing technology.

II. EVOLUTION OF COMMAND AND CONTROL SYSTEMS

Ten years ago a mobile enabled Common Operating Picture wouldn’t have been possible. Five years ago, it was possible but not practical. This section covers the evolution of Command and Control systems that were the basis for the foundation of our thesis.

A. WORLDWIDE MILITARY COMMAND AND CONTROL SYSTEM

Following the events of the Cuban Missile Crisis, the government saw a need for a system that would allow authorities to exercise command and control over U.S. forces dispersed throughout the world. The resulting system was the Worldwide Military Command and Control System (WMCCS). Its primary mission was to provide:

> A means by which the President and the Secretary of Defense can: receive warning and intelligence upon which accurate and timely decisions can be made; apply the resources of the Military Departments; and assign military missions and provide direction to the Unified and Specified Commands. (DoD, 1971)

The program required that all Department of Defense systems be configured to support the umbrella system. The final system consisted of computers, software, and communications lines forming an international network.

The use of the system, however, experienced several failures. One of the most notable occurred in June 1967 during the hostilities between Israel and Egypt. The Joint Chiefs of Staff utilized WMCCS to order the USS Liberty to move further away from Egyptian and Israeli coastline where it was performing reconnaissance. The five high-priority

messages that were sent over the system didn’t arrive for thirteen hours. As a result, 34 American sailors were killed by an Israeli attack on the ship.

Most of the problems WMCCS experienced were blamed on the system’s attempt to force existing technology to meet its specifications. Additionally, a 1979 report stated that the:

> WWMCCS ADP program was not responsive to national or local level requirements, was not reliable, lacked economical and effective growth potential, could not transfer data and information efficiently, made it extremely difficult and costly to exploit ADP technology, impaired each command’s operational backup capability, and encouraged independent and decentralized software development efforts. (World Wide Military, 1979)

The report demanded either a major overhaul of the system or a complete replacement.

In the early 1980s, WWMCCS was modernized and, despite its track record, demonstrated excellent performance in Desert Storm. Shortly following the conflict the technology the system was founded on was deemed too outdated and the decision was made to replace the system rather than try to modernize.

B. GLOBAL COMMAND AND CONTROL SYSTEM

The follow on system to WWMCCS was the Global Command and Control System (GCCS). Upon the release of GCCS in 1996 the Department of Defense boasted the introduction of modern day client-server computer architecture to replace the mainframe computers WWMCCS had been using since the 1970s. They also stated that commanders could now effectively:

> Coordinate widely dispersed units, receive accurate feedback, and execute more demanding, higher precision requirements in fast moving operations. (Global Command, 1996)

In addition, GCCS was built with the humanitarian, as well as the military, missions in mind.

GCCS has evolved over the years and several versions now exist specialized for Army, Air, Naval and Joint forces. Parallel capabilities are operated on the Non-classified Internet Protocol Routing Network, the Secret Internet Protocol Routing Network, and the Joint Worldwide Intelligence Communications System enabling collaboration between U.S. forces and our NATO allies.

Looking at GCCS today, it does not seem like a very impressive system. The maps are of poor quality and, in our experience, the system often fails to update for several days. Despite its contribution to the U.S. Military, it is once again time to upgrade to new technology.

C. BLUE FORCE TRACKING

Blue Force Tracking (BFT) is a crucial element to situational awareness. It enables leaders to track where their forces are deployed. Furthermore, it enables field entities to know their current location and the location of friendly entities in their vicinity. Historically this has been done in the field with a compass and a map. The entities' location would then be relayed back to command centers where they would be tracked on a separate map. Unfortunately this system was prone to error. Not only was it difficult to determine one's exact location using only a

compass and map, but there were often reporting errors as well as mistakes made when mapping reported locations in the command centers.

The advent of the Global Positioning System (GPS) removed a great deal of error from the equation. GPS enabled field entities to constantly know their exact location. GPS, however, did not remove the possibility of error in reporting or in command center mapping. In addition, even with GPS, command centers would only know the last reported location of field entities.

Wireless data transfer removed the remaining possibilities for error. The combination of GPS and wireless data transfer enabled units to retrieve their location and autonomously pass that location to command centers. Not only did this eliminate reporting and mapping errors, but also provided near real time tracking of entities.

Force XXI Battle Command Brigade and Below was one of the first systems to take advantage of these capabilities. The system, which initially surfaced in the late 1990s, was comprised of application software connected to GPS satellite Receivers. It utilized tactical Internet to exchange positioning data between field units and command cells (Ebbutt, 2008).

Current technology has nullified the requirement to carry a device solely for BFT needs. Most modern smartphones are enabled with both GPS and wireless data transfer capabilities. These devices are smaller, easier to use, and provide more functionality

D. GOOGLE MAPS

Google Maps is a recent mapping technology created by Where 2 Technologies, Sydney, Australia, and purchased by Google in October of 2004. It is a web-based mapping system that provides high-resolution aerial imagery over most of the world. It is primarily Javascript-based, which quickly lead to reverse-engineering to provide client-side scripts. These scripts allowed users to modify the maps with overlays, such as markers and pictures.

The development of the server side-scripts prompted Google to launch Google Map Application Programming Interface (API). The Google Map API allows web developers to embed Google Maps in their own websites. The API also provides standard Javascript procedures to allow users to customize the maps. Customizations include overlays of markers, pictures, pop-ups, drawings, and many more.

Google Map API, combined with dynamic server-side web programming, makes collaborative maps possible. After saving overlay information to a server database, other users are able to retrieve the information via the web and display the overlays locally. In order to maintain an accurate real-time display of markers, external users must repetitively access the server database to check for any changes.

SPARCCS, like most current common operational picture services, utilizes Google Maps as the application’s home-screen. Using Google Map API, users are able to place markers on the map. These markers represent missions, other users, points of interest, and pictures taken (Google, 2011).

E. NEXT GENERATION INCIDENT COMMAND SYSTEM

While a number of systems have utilized collaborative maps, one that has been getting attention lately is the Next Generation Incident Command System (NICS). Formally the Lincoln Distributed Disaster Response System, NICS is being developed by the Massachusetts Institute of Technology's Lincoln Laboratories in conjunction with California Department of Forestry and Fire Protection (CAL FIRE) and the Department of Homeland Security. NICS prides itself on being an integrated, sensing, command and control system. The system enables collaborative disaster response and interoperability to improve situational awareness. While the system is still in development, CAL FIRE began employing it in Southern California for the 2010 fire season and successfully used it in over 85% of Riverside and San Diego wildfires that season (Lincoln Laboratory, 2011).

NICS is a Google Map-based system that allows collaboration among users. In addition to user input, NICS receives sensor input from a number of deployed entities. GPS devices report location data for vehicles and personnel for display in NICS. Airborne mounted sensors relay Real-time video, images and weather to the system. The system can currently be accessed by mobile devices that employ the Firefox web-browser and is primarily accessed in the field by laptops, and tough-books with Wi-Fi or SATCOM capability. Though it can be accessed from a smartphone with Firefox, the system was not formatted for a small screen size, rendering it relatively impractical for these devices. Lincoln Laboratories has expressed interest in

developing a mobile application, but as the system itself is still in development, the mobile application is not a current priority (Next Generation, 2011).

NICS provides a good example of a PC, server-based COP. The amount of investment received by NICS, coupled with the fact that it has been utilized in both testing and actual disaster relief scenarios, suggests that it is a good case study. The NICS functionality provides a solid benchmark of necessary functionality for a successful COP tool.

F. ADVANCED GROUND INFORMATION SYSTEMS LIFERING

Advanced Ground Information Systems (AGIS) currently offers a first responder mobile application for peer-to-peer communications in the field, called LifeRing. The application offers many useful capabilities for information distribution, communication, location and GeoIntel to provide a COP amongst users. LifeRing was originally only available on Windows Mobile-based devices but has recently been extended to include most mobile operating systems. LifeRing’s key functionality includes the ability to exchange images and video, Blue Force Tracker, and collaboration between multiple devices utilizing different operating systems.

While LifeRing is a useful system for mobile devices, it has some issues that SPARCCS aims to address. The first issue is the maps on the mobile devices. While the PC version of LifeRing can access and utilize Google Maps, the mobile version cannot. The maps on the mobile devices are preloaded which use critically limited storage space. In addition, the maps on the mobile devices are poor quality

and do not have satellite capability. SPARCCS intends to use Google Maps on both its mobile and headquarters version of the application.

Another issue is the way devices are connected to each other. LifeRing is server-based while SPARCCS aims to take advantage of distributed computing combined with cloud computing for its server and database needs. By doing this SPARCCS takes advantage of all the capabilities cloud computing has to offer, especially that of disbursed data locations. This insures that an incident or disaster at a server location is unable to bring down the system. In addition, SPARCCS will have the ability to turn a mobile device into a server for a unit or team. This should not only decrease bandwidth usage but also insure that if the server should fail, the unit or team will be able to continue using the system (AGIS, 2010).

G. CONCLUSION

In conclusion this chapter summarizes the evolution of predominant Command and Control Systems and how their evolution has made it possible to design current Common Operating Picture programs. The next chapter will discuss existing technology that we used to make Common Operating Picture system SPARCCS.

III. INFRASTRUCTURE BACKGROUND

This chapter will discuss prevailing technology that makes the SPARCCS Common Operating Picture system different from other systems on the market today. This will include cloud computing, not so relational databases, Google App Engine, and Web Services.

A. CLOUD COMPUTING

1. Definition

Cloud Computing can be defined as a model for enabling convenient, on-demand network access to a shared pool of dynamically scalable, and often abstracted, computing resources. Though the basic concept of this form of computing may not be groundbreaking, its re-emergence since 2008 has introduced new possibilities to satisfy a nearly insatiable need for computing power and memory. For example, a Cloud deployment model can offer users and developers the option of utilizing multiple servers or storage devices that appear as one logical resource. This dramatically increases the amount of physical drive space and sheer computing power available without the need to invest in a local hardware infrastructure. Solving larger and more complex problems in shorter amounts of time, while spending ultimately less money to do so, is a possibility that has greatly piqued the interest of international corporations and government agencies alike.

2. Cloud Service Models

Cloud computing has been utilized in very concrete forms already. A well-known example of this application

comes in the form of SaaS, or Software as a Service, model. In this case, on-demand software is made available to users remotely via the Cloud, typically through a web application or thin client application. Very popular forms of this software delivery system, such as Google Docs or the Zoho Office Suite, offer users the ability to collaborate and complete projects with greater ease and efficiency.

Another well-developed Cloud service model is known as IaaS, or Infrastructure as a Service, in which consumers are able to provision fundamental computing resources to run arbitrary software. For example, if the owners of a small business didn’t have the infrastructure necessary to run a batch job in an acceptable time frame, they could utilize Amazon EC2’s elastic computing capacity to get it done for a relatively small price.

Finally, there exists a third model, known as PaaS, or Platform as a Service. In this case, developers can upload web apps they create specifically for a Cloud environment. The most prominent example of this is the Google App Engine, in which developers upload web apps published via the Google App Toolkit. The Google App Engine and Toolkit will be further explained, as this is the backbone for the thesis work (Tech Target Sites, 2007).

3. Implementation

For a developer trying to make full use of the Cloud’s offerings, a clear implementation decision needs to be made between a thin client application and a web-browser application. Thin client architecture implies an operating system exists locally, but it is allowed very few independencies from the Cloud. The application must connect

to the Cloud and relies heavily upon the Cloud services for computing and storage. On the other hand, web-browser applications can also be used to interact with the Cloud environment; the difference being that a web-browser application in a Cloud environment simply uses the web-browser as the front-end layer where the user interaction takes place. The resulting interaction accesses the back-end layer of the application where the hardware and software architecture exist on the Cloud. For this reason, a web-browser Cloud application is inherently operating system agnostic, which allows potential users to be free of certain system restrictions. This is typically how a popular Cloud program, like Google Docs and Apple MobileMe, functions. Certainly, both implementations offer their own benefits and drawbacks. This thesis will focus on the web-browser based application due to its operating system independence.

4. Benefits of Cloud Computing

Cloud computing offers several potential benefits over more traditional host-oriented or client-server computing approaches. Some of these are identified below.

- Cost Reduction
    - "Federal agencies could eventually save 85 percent of their yearly information technology infrastructure budgets by moving operations to either a public, private or hybrid cloud." (Booze Allen Hamilton, 2010)
    - "An agency would spend about $77.3 million to run 1,000 servers annually in-house, but it would

cost only $22.5 million to run them virtually in public cloud environment, $28.8 million in a hybrid cloud—a mixture of public and private cloud-based services—or $31.1 million in a private, agency-run cloud." (Booze Allen Hamilton, 2010)

- Flexibility
    - "Adjust cloud-based resources up and down to meet real-time needs, or offload onsite data to the cloud as needed to improve operational efficiencies. And since the cloud is Internet-based, you can access these resources from anywhere, supporting remote work and continuity of operations." (Microsoft, 2011)
- Collaboration
    - "With both the application and the data stored in the cloud, it becomes easy for multiple users—located anywhere in the world—to work together on the same project." (Microsoft, 2011)
- Disaster Recovery/Continuity of Operations
    - "Centralized data storage, management, and backups, data recovery in response to local business disruptions can be faster and easier." (Microsoft, 2011)
- Applications and content
    - "Rather than waiting in the software procurement line, hosted software, datasets, and services are

available as they are released" so the mission can be the focus instead of the infrastructure (Microsoft, 2011).

- Creative IT
    - "Since cloud services can be centrally managed, IT workers are freed from a "keep-the-lights-on" approach, providing more time to foster creative problem solving." (Microsoft, 2011)

5. Cloud Computing Conclusion

By utilizing a web-based cloud application, information can be easily accessed when needed. The user only needs have access to the Internet to retrieve and update information. Other benefits of the cloud include lowering costs as it pools computing resources that can be distributed to applications as needed, responsive deployment of the application as there is no need to update all networking hardware and software within a given network infrastructure, and scalability. Most importantly, it has redundancy in the case that the crisis arises in the area of the cloud disaster recovery services can bring up a separate server immediately.

B. DATABASES

In order to understand the full service of cloud computing an understanding of databases must be gained to appreciate the intrinsic utility of working within a cloud environment.

1. History

With computer use finally entering the corporate world in the 1960s, large amounts of data needed to be stored and organized. Companies typically had many concurrent end-users all dealing with a large amount of diverse data. Databases were formed to help ease the increasing difficulties in designing, building, and maintaining complex information systems. The formation of databases coincided with the availability of direct-access storage. This was in stark contrast with tape-based storage systems used up until that point. Databases allowed for share d interactive usage of data for multiple users ("Databases," n.d.).

In the earliest database systems, accessing data efficiently was the primary concern. In order to do so the key aim was to make data independent of the software logic of the applications. This made it so that the same data could be made available to different applications at the same time. The first generation of the databases sought to address these issues. It was known as a navigational database. Applications using these kinds of databases had information linked together through a series of pointers that lead from one series of records to another ("Databases," n.d.).

The Relational model, first proposed in 1970, departed from this tradition by requiring that applications should search for data based on the content of the data. This was considered necessary to allow the content of the database to evolve without constant rewriting of the applications to deal with evolutions in the data model. The relational

database holds data in tables with a fixed structure of rows and columns (“Databases,” n.d.).

The relational database, still the most popular database, too, has its limitations. The rigidity in its structure alone has been seen as weakness where all data must fit into a pre-determined schema of tables allowing for little flexibility to change on the fly. Relational databases also have a weak ability to handle information that is more varied in structure. For example, document databases, engineering databases, multimedia databases or databases used to store information for molecular sciences all have trouble when forced to fit into the classical model (“Databases,” n.d.).

2. Problems with Relational Databases within the Cloud Environment

a. Scalability

One of the big pushes in today’s computing environment is to move information to the cloud. The cloud has distinct advantages in information storage and cost. However, moving the relational database to the cloud has its share of difficulties and drawbacks.

Relational databases do have the ability to scale well; nonetheless, this scalability is normally planned and done over considerable time. Yet, for cloud storage purposes, the ability to provision and decommission servers on demand, referred to as dynamic scalability, is needed. This is because workloads can triple overnight and then decrease to half the storage needed from where the applications started. Classic relational databases are simply not designed for this sort of environment, as there

must already be the hardware infrastructure in place to handle the maximum capacity of users or data, even if it means the hardware goes unused for a majority of the time (Howard, 2011).

b. Mapping

The second issue that relational databases face with respect to moving to the cloud is that they are a poor fit for most software development. Current computer programming has changed over the past fifty years from that of “spaghetti code” to a much more organized, object-oriented approach. In this new approach, object-to-relational mapping (and vice versa) needs to take place between the interface, the database and the software logic. This mapping adds complexity and cost while decreasing performance if a traditional database is used (Howard, 2011).

3. Not-so-Relational Databases: A Solution for Cloud Computing

Not-so-Relational (NSR) Databases have emerged to bridge the gap between cloud data storage and object-oriented programming. They do not suffer from the same complexity and cost issues because they store information in a more object-oriented fashion, using key/value pairs (Howard, 2011).

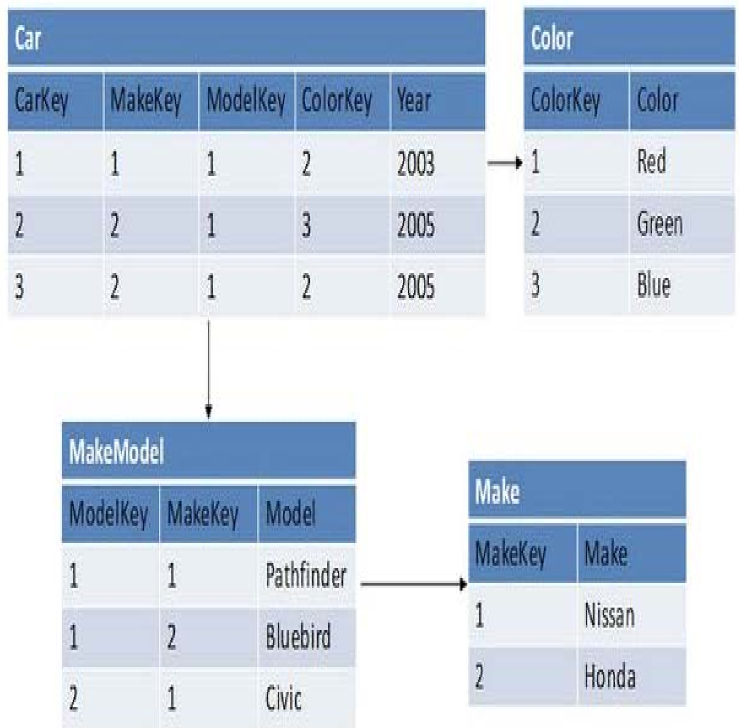
Key/value items store all related information about a single item or object as a single entity, as opposed to having multiple tables in relation database linked by primary/foreign key relationships. This eliminates the joins necessary in conventional database environments. The

items themselves are stored in a table, also known as a domain. Completely different types of items can all be stored in the same domain. To inter-relate different items, keys are used. If two distinct items have the same value for a particular key then they are related (Howard, 2011).

4. Key Differences between Relational and Not-so-Relational Databases

a. Definition

Relational databases contain tables, tables contain columns and rows, and rows are made up of column values. Rows within a table all have the same schema. The data model is well defined in advance. A schema is strongly typed and it has constraints and relationships that enforce data integrity. The data model is based on a “natural” representation of the data it contains and not on application functionality. The data model is normalized to remove data duplication. This normalization establishes table relationships, and these relationships associate data between tables (Bain, 2009). Figure 1 displays a simple relational database.



Example of a Typical Relational Data Model

Figure 1. Relational Database

In NSR databases, domains can initially be thought of like a table; but unlike a table, you do not have to define any schema for a domain. A domain is basically a bucket that you put items into. Items within a single domain can have differing schemas. Keys identify items, and a given item can have a dynamic set of attributes attached to it. In some implementations, attributes are all of a string type. In other implementations, attributed have simple types that reflect code types, such as integers, string arrays, and lists. No relationships are explicitly defined between domains or within a given domain. Figure 2 shows the same information in a much simpler NSR database. (Bain, 2009)

Car	
Key	Attributes
1	Make: Nissan Model: Pathfinder Color: Green Year: 2003
2	Make: Nissan Model: Pathfinder Color: Blue Color: Green Year:2005 Transmission: Auto

Figure 2. Not-so-Relational Database

b. Data Access

In relational databases data is updated, created, deleted, and retrieved using Structured Query Language (SQL). SQL queries can access data from a single table or multiple tables, the latter through table "joins." SQL queries include functions for aggregation and complex filtering. They usually contain a means of embedding logic close to data storage, such as triggers, stored procedures, and functions (Bain, 2009).

In NSR databases data is created, updated, deleted and retrieved using API method calls. Some implementations provide basic SQL-like syntax for filtering

criteria, and basic filter predicates can often only be applied. All application and data integrity logic is contained in the application code (Bain, 2009).

c. Application Interface

Relational databases tend to have their own APIs, or make use of generic API, such as OLE-DB or ODBC. Data is stored in a format that represents it natural structure, so it must be mapped between the application code structure and the relational structure. This can be seen from the Figure 1. In this example, there is a complicated mapping and joining between tables for information about the cars in the database. This would in turn require helper classes within the program code to get all of the information out of the database required for a specific car then store it as application data (classes) to be manipulated in the program (Bain, 2009).

NSR databases tend to provide Simple Object Access Protocol, Web, and/or Representational State Transfer API's over which data-access calls can be made. Data can be more effectively stored in application code. Entire classes and their components can be sent between the database and the application, instead of merely strings that have to be parsed and then refactored as the application code dictates. This allows for database programming and management that is compatible with it the application's object oriented structure. With this new scheme, only communicative relational "plumbing" code is needed for database application interaction. This is demonstrated in Figure 2. In this example, the structure of the database resembles that of a class with simple data

member objects that can easily be mapped to the application code to a car class. Note the difference from Figure 1 in simplicity (Bain, 2009).

5. Google Big Table and Datastore

“Google’s BigTable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in the BigTable, including web indexing, Google Earth, Google Finance and Google App Engine… The BigTable is a sparse, distributed, persistent, multidimensional stored map (Fay Chang, 2006).” A row key, column key, and a time stamp form the index to the map.

The piece of the BigTable used for comparison will be the Google Datastore, Google’s response to Amazon Simple DB. The Datastore is built on top of the BigTable, and acts as an internal storage system for handling structured data. In and of itself, the Datastore is not a direct access mechanism to the BigTable, but can be thought of as a simplified interface on top of the BigTable. It “gives clients dynamic control over data layout and format (Fay Chang, 2006).”

The Google Datastore focuses on flexibility of data. In the Datastore objects are known as entities. And an entity can have one or more properties. A distinguishing feature from traditional databases is that these properties can be more than only simple data of Strings, Integers, and Booleans, but they can also be classes and even references to other entities or classes. So if their needs to be a

change in the data model no changes to rows or columns need to be made only a change to the entity (Google, Datastore Overview, 2011).

a. Benefits of the Datastore

The Datastore can perform numerous processes in a solitary atomic operation. Within the Datastore model, an operation cannot be considered successful unless every process is completed. If any singular process fails or is disrupted then the operation is terminated. In order to deal with race conditions the Datastore uses an “optimistic concurrency” model. Within this archetype, the first application instance to modify an entity has control over that entity’s data. All subsequent application instances trying to manipulate said entity’s data will fail to submit their changes until the initial application instance has completed its transaction. This feature is especially helpful when operating within a distributed network architecture with multiple users trying to use or access the same data. Likewise, the Datastore uses a distributed architecture to automatically manage the scaling of large amounts of data, known as the BigTable, described above. Further, by using the Datastore or in effect any large company’s computing resources, a client understands that they are receiving a reputable, secure, and reliable product. This is especially important in the information technology world where it is important to deal with trustworthy companies for data storage (Google, Why App Engine, 2011).

Now with an understanding of databases, Google App Engine and Google Web Toolkit can be presented to show how we took advantage of cloud computing with Not-So-Relational Databases.

C. GOOGLE APP ENGINE AND GOOGLE WEB TOOLKIT

1. Google App Engine

As previously mentioned, the Google App Engine (GAE) is a platform for developing and hosting web applications in Google managed data centers. First released in April 2008, GAE uses Cloud computing technology to virtualize applications across multiple servers. The backend piece is the Google BigTable entitling users to the advantages of the Google Cloud. The app engine is a free service for users requiring less than 1 GB of storage and less than 5 million page views per month. The App Engine provides dynamic web serving, persistent storage, automatic scaling, APIs, and task queues for performing work outside of a web request. It also supports common web technology and has a development environment that simulates the Google App Engine on a developer’s computer.

Applications running in Google App Engine run in a secure environment, referred to as “The Sandbox,” which controls access to the underlying Operating System. In order to maintain the environment as secure, The Sandbox places a few limitations on the application. The first limitation is that an application can only access other computers through URL and e-mail. Conversely, other computers can only connect to the application through HTTP requests. Secondly, applications cannot write to the file system in any of the runtime environments. The only files

an application can read are those uploaded with the application code. Finally, an application's code can only be run when a web request is made, or as a queued or scheduled task.

The App Engine features a Java runtime environment enabling users to build applications using standard Java technologies, such as the Java programming language. The App Engine also provides a distributed data storage service. This service is utilized to store SPARCCS data. The cloud-based application manages HTTP requests from mobile devices (Google, Why App Engine, 2011).

2. Google Web Toolkit

Google Web Toolkit (GWT) provides a developer environment for common object-oriented code to be repurposed and optimized into web-browser-independent applications (Google, 2012). The toolkit is composed of four elements: the software development kit (SDK), speed tracer, Eclipse plug-in, and the interface designer. Exploring each one individually will give an overview of the application development process using GWT.

a. Software Development Kit

The GWT SDK is the backbone for writing web based cloud applications. Most of the time spent developing software is not the actual programming of the code but in debugging and maintenance. This becomes exasperated when developing cloud web applications, as each individual workstation can run multiple operating systems and multiple web browsers, leaving the developer spending a majority of time modifying code to work in different environments. The

GWT SDK affords the user the ability to use high level programming languages, such as Java or Python, which are compiled into JavaScript in such a manner as to increase the computing speed of the program, in order to create applications that can be run on any browser, on any operating system. The developer no longer has to worry about converting code to work under the variety of different operating systems, which is a circumstance that readily occurs in a network centric system.

Furthermore, the SDK is as intuitive as using the normal software development process for any desktop-based application, as you can use the same “edit-refresh-view” cycle to see how the application is running on a development server. This process is normally not allowable with most web-based application building, since the program has to be compiled then sent to the server before it can be viewed in its entirety. Using the process enabled by GWT, errors can be caught during development instead of at run time. Coupled with J-Unit testing integrated from normal Java code testing, developers save time throughout the writing and debugging process. GWT’s intuitiveness comes from a complier, which accomplishes a complete examination of the Java source code, eliminating dead code and trimming classes that are not used. As a result, the compiled code is smaller and runs faster than if run from straight JavaScript. The SDK also allows developers to abstract themselves from low-level networking protocols. This makes cloud application programming easier and more accessible to the average programmer. Being well versed in XMLHttpRequests (XHR) calls used for client server

communication are no longer an issue because the GWT innately handles the client-server communications (Google, 2012).

b. Speed Tracer

GWT's Speed Tracer is the next step in optimization for applications running in a cloud computing system. Speed tracer is a development tool to aid in the recognition and resolution of performance issues in applications. It uses visual standards that are taken from low-level points of the browser and analyzes them. The information is displayed on a browser page using visual charts and graphs as well as tabled statistical data that is easy to understand and compare. By using the Speed Tracer the developer can get an enhanced look at what the application is doing during run time. With this information an overall picture can be developed on how much time the application spends doing each operation, allowing the developer to delve into optimized programming methods for speedup across platforms. "Problems caused by JavaScript parsing, layout, CSS style recalculation and selector matching, DOM event handling, network resource loading, timer fires, XMLHttpRequest callbacks and painting" can easily be found and rectified enriching the overall cloud computing experience (Google, 2012).

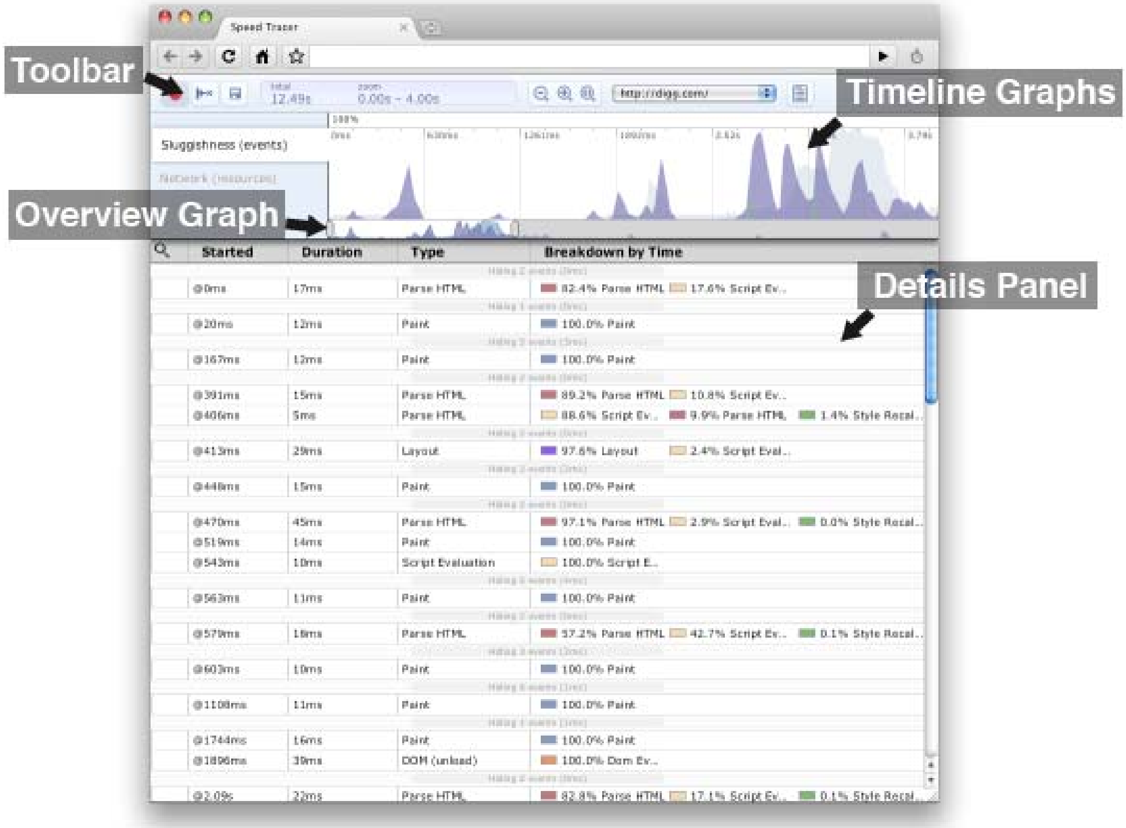


Figure 3. Google Speed Tracer

Figure 3 is a screen capture of the speed tracer in progress. The important portions are noted with the labels. The toolbar allows the user to record reset and save information, as well as view and manipulate the graph data. The timeline graph allows the user to see information as it happens in real time as well as analyze network and program data from the past. The overview graph displays all of the network information about the program since its last post to the cloud. The details panel is the most helpful, as it goes into detail about specific events happening in the program. Most notably is the hints section, which helps

recognize common errors in the application and helps provide common software engineering solutions if possible (Google, 2012).

c. Eclipse Plug-in

GWT is simply packaged in an easy to install and use Eclipse plug-in. This allows the developer to use all the tools of the toolbox bundled in an IDE instead of hand-writing code, terminal compiling, loading the code to a separate server, then accessing the server to run the application. By using the IDE, the developer is already in an intuitive programming environment that has been well established with technical support. Now code can be quickly designed in an IDE instead of working in tedious terminal windows and text editors with no specific code hierarchy. This allows the developer to focus on application soundness and more efficient code generation (Google, 2012).

d. User Interface Designer

The final piece of the GWT tool kit is the User Interface Designer (UID). It allows for bi-directional Java user interface design, meaning that the program’s user interface can be created and source code populated or source code can be written then viewed in the designer. This drastically reduces the time expended creating user interfaces, since the developer can use “drag-and-drop” techniques to place design items exactly where the designer wants without having to hardcode all of the elements. Moreover, all code, not just special portions of code, can be edited in the source code if desired. The UID also allows for the custom creation of panels and composites

that can be packaged and reused in future projects, instead of being restricted to predesigned modules (Google, 2012). Although this capability of the GWT UID is available to all GWT users as a separate plug-in, it was not used in this thesis.

e. Conclusion

We believe that PaaS is the future of Cloud Computing. As a developer trying to make full use of the Cloud’s offerings in a competitive environment, we think that the Google App Engine is a suitable development environment because it is robust, affordable, and highly scalable platform for implementing the cloud based web application for our thesis.

Android is the next piece of the system architecture; it runs on the smartphones used in the SPARCC’s environment.

D. ANDROID OPERATING SYSTEM

Android is a Mobile Operating System (OS) developed by the Open Handset Alliance and purchased by Google in 2005. It has grown in popularity in recent years, reaching a 36% market share of smartphones at the end of the first quarter in 2011 (Meyer, 2011). The popularity of the device is of extreme significance because it means many responders will already have them and are familiar with its user interface and functionality. Furthermore, U.S. officials, and the military as a whole, have plans to receive Android devices capable of handling classified material this year (Milian, 2012).

There are many key components of Android with relation to developing a collaborative-map mobile application. For instance, a camera and microphone, coupled with media support for common audio, video, and still-image formats, provide a variety of methods for information input. In addition, GPS and accelerometer hardware can provide important data concerning the location and current action of the user carrying the device. Depending on the device and location, Android supports Bluetooth, EDGE, 3G, 4G, GSM, and Wi-Fi, providing a variety of means for sharing collected information in the field. Likewise, it is multi-channeled and on multiple carriers. This allows for a greater range of use depending on the cell towers in the area in which the devices are being used.

The Android OS comes with an embedded database called SQLite. SQLite distinguishes itself from other databases in the sense that it does not have a separate server process; it reads and writes directly to disk files, where a complete database is stored in a single disk file. Using SQLite we are able to sync SPARCCS information on the phone with information uploaded to the cloud.

Compared to other operating systems, Android offers the most user-friendly development process. As evidenced by Angelos Stavrou, “the government chose to work on Android first because Google already allows people to tinker freely with its code” (Milian, 2012). Mr. Stvrou is the information-security director at George Mason University who is working on the government project to bring Android smartphones to the U.S. military. He is further quoted saying “Android was more cooperative in supporting some of

the capabilities that we wanted to support in the operating system, whereas Apple was more averse" (Milian, 2012). Likewise, all Android applications are programmed in the Java programming language and developers are able to use device hardware, access location information, run background services, set alarms, add notifications, and more. Developers have access to all the APIs used by the core applications. In addition to Java, Android includes a set of C/C++ libraries used by various components of the system. Programming takes place in the Eclipse IDE, while code-execution can take place on either a device emulator or an actual device. Unlike iPhone, Android does not require developers to pay an annual fee.

All in all, Android Operating System is a very capable smartphone system, which allows for the customization and security considerations that fit military smartphone application development. The smartphones using the Android Operating System and the web-based cloud application communication communicate via HTTP Web Services, described in the following section.

E. WEB SERVICES

The latest definition of Web Services, as defined by W3C Working Group, is "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine—processable format (specifically WSDL—expand WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards" (W3C Working

Group, 2004). In simple terms, it is how two different software applications may communicate with each other across the Internet. Web Services are not concrete pieces of software; instead, they are abstractions or agents, the actual pieces of hardware and software and the acting forces behind web services. In order for two agents to be able to communicate they must agree upon a web service description or WSD, and web service semantics. “They define the message formats, data types, transport protocols, and transport serialization formats that should be used between the requester agent and the provider agent…[as well as] the shared expectation about the behavior of the service” (W3C Working Group, 2004). The engagement between a web service provider and a web service requester is detailed in Figure 2. When both parties become known to each other they can agree on web semantics and definition. Then, through a request and provider agents, the two parties are able to communicate with each other across the Internet.

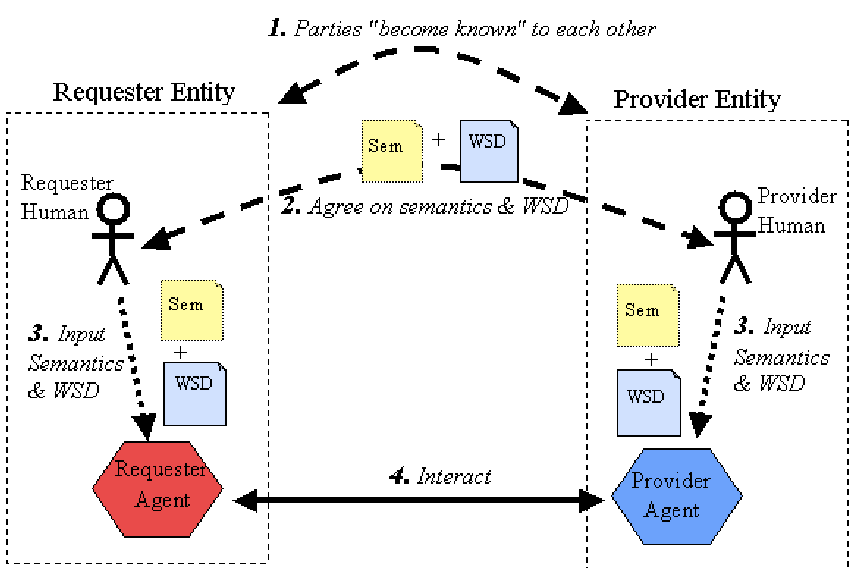


Figure 4. Web Services

For the purposes of the SPARCCS project we chose to implement the semantics and definitions followed under simple HTTPPosts and HTTPRequests using HTTP Servlets for Android to server communications and Remote Procedure Calls for web-client to server communications. This is because these methods were proven services that would be able to transfer both text and media to and from the database on the server.

F. SUMMARY

This chapter details the role of cloud computing and Not-So-Relational Databases as the backbone structure for the SPARCCS infrastructure. It shows how this new technology is more flexible and versatile than the old client-server model using traditional relational databases. Furthermore, using the cloud model allows for a sounder software engineering approach to our system of systems. Within the architected model we take advantage of, the openness and versatility of the Android operating system, the reliability and robustness of the Google App Engine, and the simplicity and consistency of HTTP web services to create the SPARCCS system. The next chapter will detail our system, with an overview of key features and software engineering highlights to fully show the development process.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SPARCCS IMPLEMENTATION

Having discussed the history of Common Operational Picture systems, as well as the technology utilized in our Smartphone Assisted Readiness, Command, and Control System, we will now explain the functionality of the system and proper use. The first section of this chapter will describe a sample scenario, followed by key features of SPARCCS and then the overall system architecture. The following chapters will go into detail of both the web browser based, and Android based systems, respectively. For visualization convenience we have provided several screen shots of the various screens a user will encounter using either system.

A. OVERVIEW

1. A Sample Scenario

The implementation of SPARCCS focuses on creating a COP that can be used by those in a central command post as well as members responding to circumstances in the field. A sample scenario for SPARCCS is as follows:

A crisis situation or military mission has arisen and several teams have been sent into the field at once to respond. Each member of the corresponding teams has an Android smartphone running the SPARCCS mobile application. The team members are capturing information relevant to their situation and sending their information from their smartphones to their team leaders' phone. The leaders of these teams gather the information and send relevant information back to the team members, if needed, and to the SPARCCS command center cloud application. This information

feed forms the basis of situational awareness in the command center. The command center assembles the feeds from the numerous group leaders and displays them for the decision makers to see what is going on in the field, and sends back relative information to the team leaders to disseminate to their teams. In doing this, the central command as well as the operational teams can be more time-efficient and better coordinate limited resources.

2. Key Features

SPARCCS implements several key features making it a robust, practical system. These features are as follows:

- Login requirement to prevent unauthorized access to the system.
- The ability to create, join, edit, and view missions.
- The ability to create, edit, view, and delete points of interest.
- The ability to capture, edit, view, and delete images.
- The ability to view all missions, points of interest, and images on a map.
- The ability to view all mission, points of interest, and responder information in a list format.
- The ability to retrieve smartphone GPS location for the placement of missions, points of interest, and images.

- The ability to store all relevant information in the Android’s local SQLite database, or the cloud database.
- Syncing between the Android and cloud databases.

These functionalities will be discussed in depth in the following sections of this chapter.

3. Architecture

The SPARCCS program consists of over fifty classes and over 75,000 lines of code. Although it is a large program it is a fairly simple architecture in implementation. The system centers around four main entity classes: Responders, Missions, Points of Interest and Images.

- The Responder class contains information about individuals reacting to operations around the world, as well as those operators working within the central command. The Responder class consist of the following notable data members: Id, First Name, Last Name, Middle Initial, Login, Password, Type (Military, Fire, Medical, Humanitarian Assistance or Law Enforcement), Unit, Phone Number, Email Address, IP Address, Associated Mission, Latitude and Longitude.
- The Mission class contains information about an operation responders are reacting to around the world. The Mission class consists of the following notable data members: Id, Name, Description, Start Date, End Date, Latitude, Longitude, Mission Leader, Mission Creator, and Miscellaneous Information.

- The Point of Interest class contains information about a particular place, person or item of significance that corresponds with an associated mission. The Point of Interest class consists of the following notable data members: Id, Name, Description, Time, Creator, Correlating Mission, Location Notes, Latitude and Longitude.
- The Image class contains a picture and corresponding information for that picture. The Image class consists of the following notable data members: Id, Name, Description, Creator, Time, Correlating Mission, Correlating Point of Interest, Location Notes, Latitude and Longitude.

Theses classes can be viewed individually in a list format or can be plotted on a Google Map. Instances of these classes are saved on a Google Cloud Database that is accessed from both the cloud application, and the Android mobile application. Using CRUD (Create, Read, Update and Delete) operations through HTTP Servlets on the Android client and Remote Procedure Calls on the web client, instances of these classes can be manipulated.

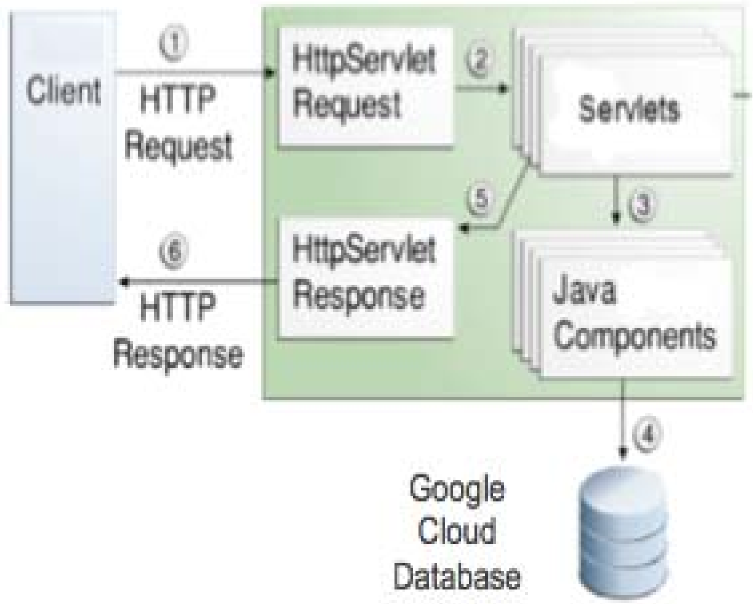


Figure 5. Android System Architecture

Figure 5 represents the overall system architecture from the Android perspective. The client is the Android mobile application. The client can request/send class information via an HTTP Request. (1) The HTTP request interacts with the HTTP servlets using HttpServlet Requests. (2) These servlets form Java components in our case Objectified Data objects (3) to perform CRUD operations with cloud database also known as the Google Datastore explained in Chapter III. (4) The servlets then create the appropriate HTTP Servlet Response (5) back to the requesting client. (6) The information is then displayed in the client.

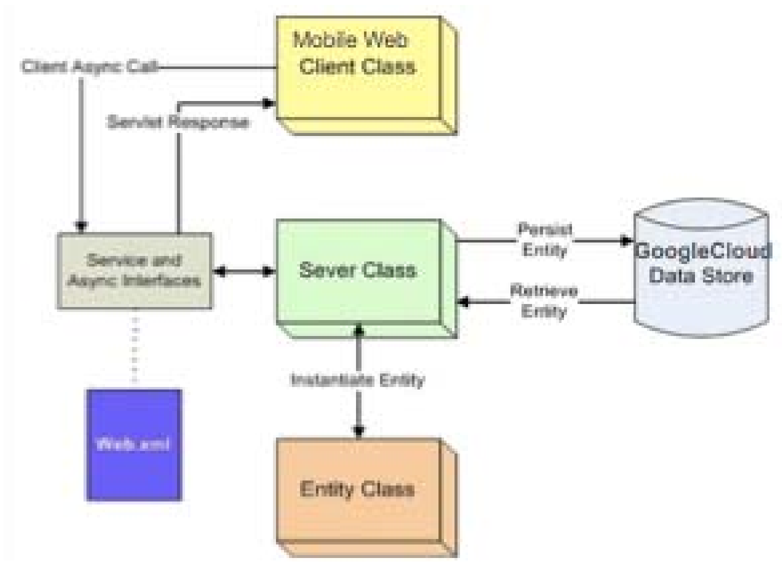


Figure 6. Web Client System Architecture

Figure 6 represents the overall system architecture for the web client perspective. The only difference being remote procedure calls are used instead of the HTTP posts and requests to retrieve and send information from the Cloud Database.

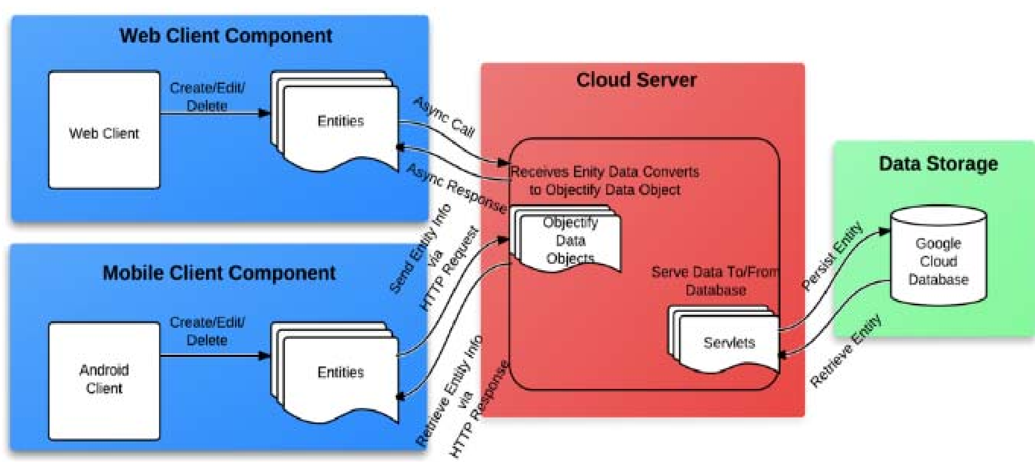


Figure 7. System Architecture

Figure 7 represents the full system architecture. It demonstrates the flow between both client applications and how they interact with the server to retrieve and send information in the entity classes.

The following sections of this chapter will further detail the workings of each client application as it interacts within the overall SPARCCS architecture.

B. CLIENT IMPLEMENTATION FLOWS

1. Cloud Application

a. Login Screen

The cloud application begins on the login screen shown in Figure 8. From here the user can log into the application or register as a new SPARCCS user. If the user decides to register as a new user they must put in all of their responder information and they will be given a user name consisting of their first initial, middle initial and last name. If there are matching names the login string is extended by a number based on when they registered.

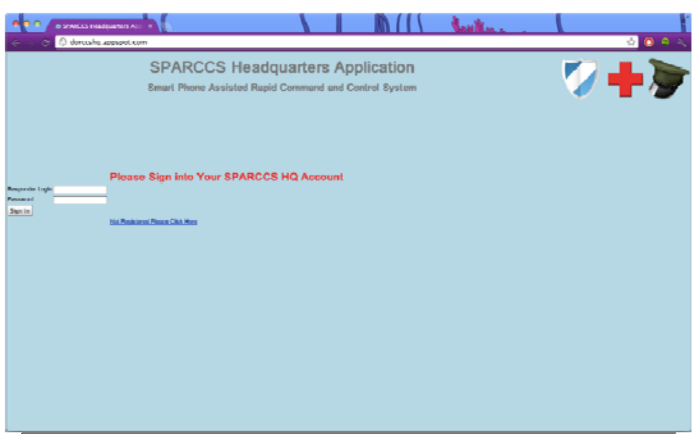


Figure 8. Login Screen

b. Initial Screen

Once the user is logged-in they are taken to the main interaction screen shown in Figure 9. On the large right hand panel of the screen is the user map. It is centered on the user’s current location. Also displayed on the map are all the current missions, points of interest, responders, and images. On the left panel of the screen is a navigation panel that opens to a list of all of the missions. The navigation panel also contains other panels for further information that we will detail later in this section.

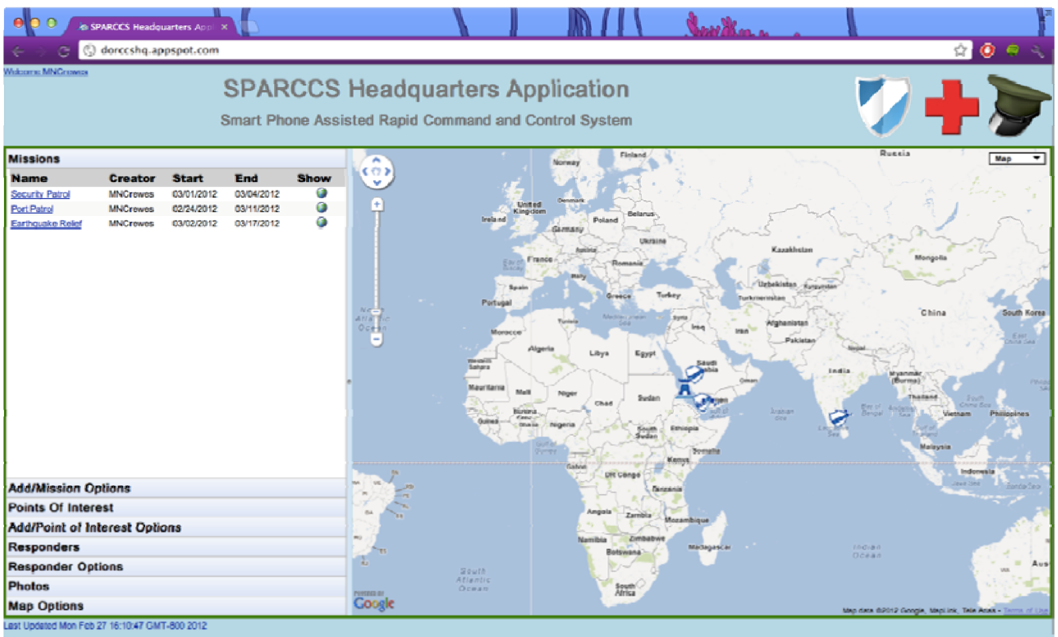


Figure 9. Main Interaction Screen

On the top of the screen is an anchor that has the user login. If this anchor is clicked information about the current logged-in user is displayed, as shown in Figure 10. From this panel the user can edit their information as

well as logout of the program. At the bottom of the screen is another anchor containing a timestamp for the last time the client was updated. If this anchor is clicked the application will update itself and the timestamp will change accordingly.

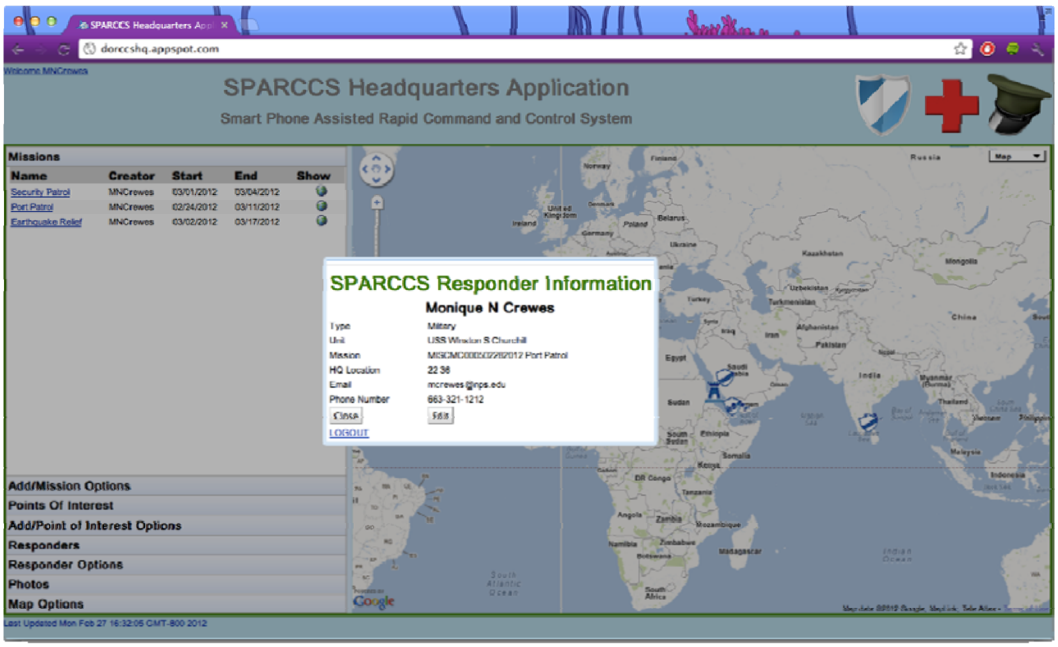


Figure 10. Responder information

From the main screen the user can choose to interact with the map directly or interact with each of the navigation panes. Interacting with either the map or the navigation panes will update the opposite interaction tool so the user does not have to worry about inputting information twice. Next we will go through all the panels within the navigation pane and describe how each pane interacts with the map and how the map interacts with the navigation panes.

c. Missions Panel

The Missions panel contains a list of all of the missions contained in the application. The initial information that can be seen is the name, creator, start date and end date. The creator of the mission is the responder’s login to keep the names short for the display. However, if the creator’s login is “moused-over” then the full name of the mission creator will be displayed in a small pop-up panel. The mission name is an anchor. If this anchor is clicked then the mission’s description will show up below the mission’s information along with a second anchor that says “more…” If this anchor is clicked then all the mission information will be displayed along with an option to edit or remove the mission if the currently logged-in user is the creator of the mission. This sequence is displayed in Figure 11.

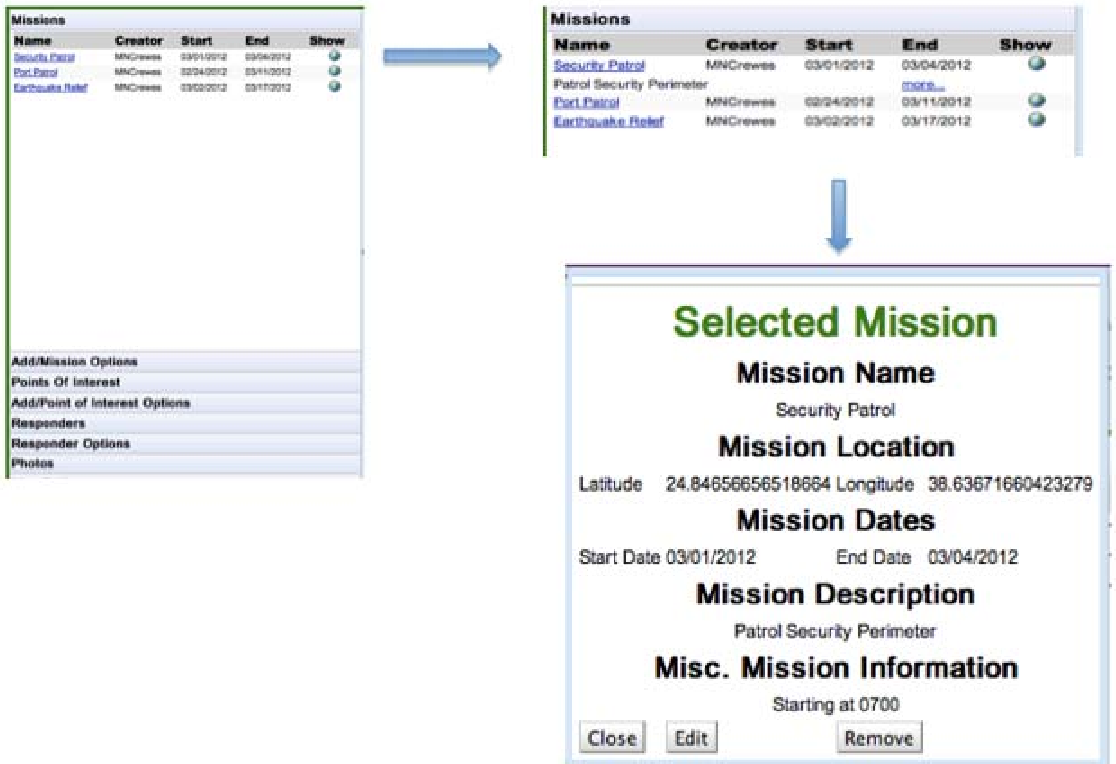


Figure 11. Display Mission Sequence

There is also a globe button associated with each mission in the list. If this button is pressed then the mission along with essential mission information will be displayed on the map, as depicted in Figure 12. If the bubble on the map is expanded then all of the mission information will be displayed, as shown in Figure 13. Within this expanded bubble are links to see corresponding responders, points of interests, and pictures on the map at the same time. Once these associations are displayed on the map the user can click on them to get their corresponding information, as seen in Figure 14.

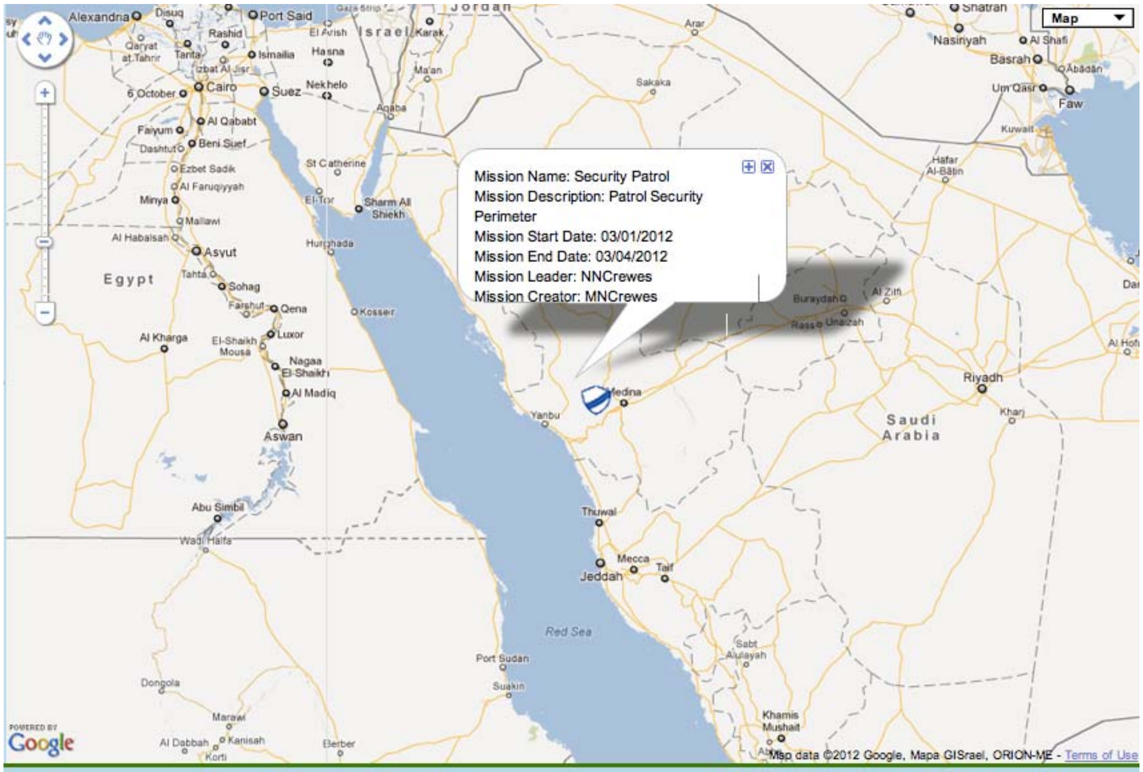


Figure 12. Initial Mission on Map

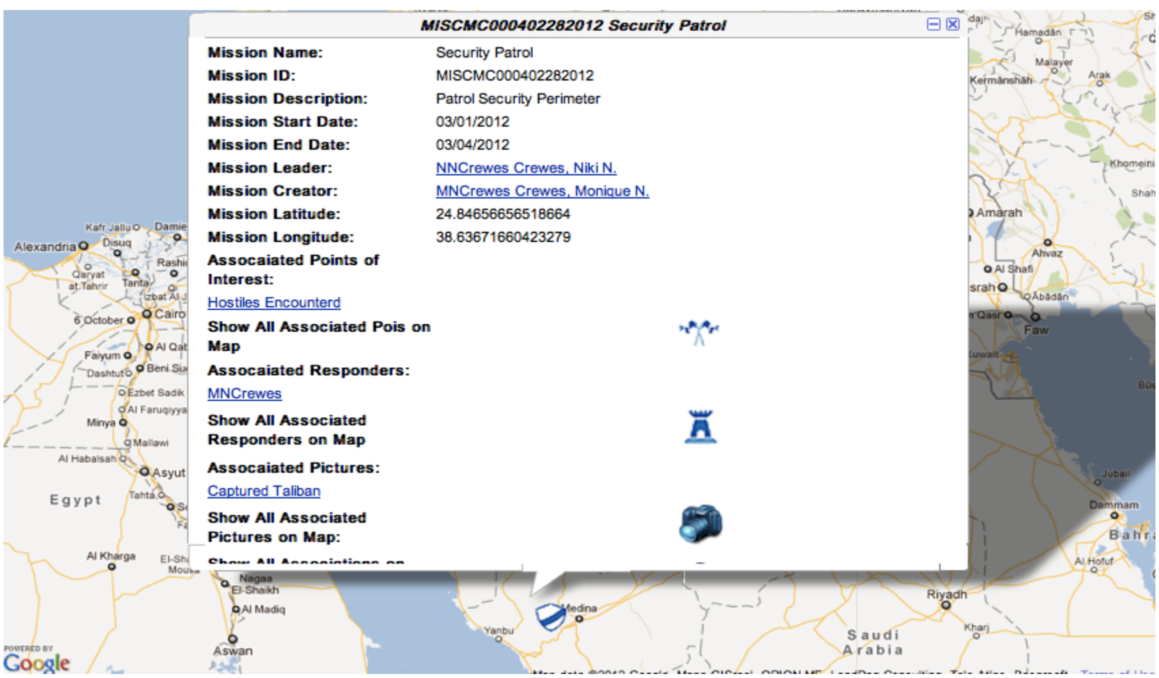


Figure 13. Expanded Mission Information

Figure 14. Mission Associations on Map

d. *Add/Mission Options Panel*

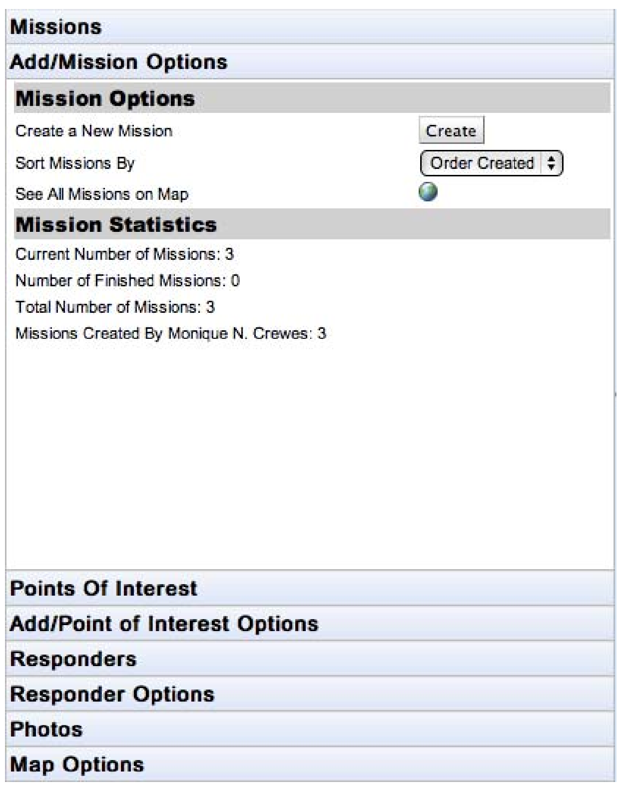


Figure 15. Add/Mission Options Panel

The Add/Mission Options Panel shown in Figure 15 allows the user to manually add a mission to the list, provides mission list display options, shows all missions on the map, and shows mission statistics. From this panel the first option is a button for the user to add a mission. Clicking this button brings up a mission form for the user to input mission information, as displayed in Figure 16. The mission must have a name, valid location, and start date (the end date does not have to be known at the time of creating the mission).

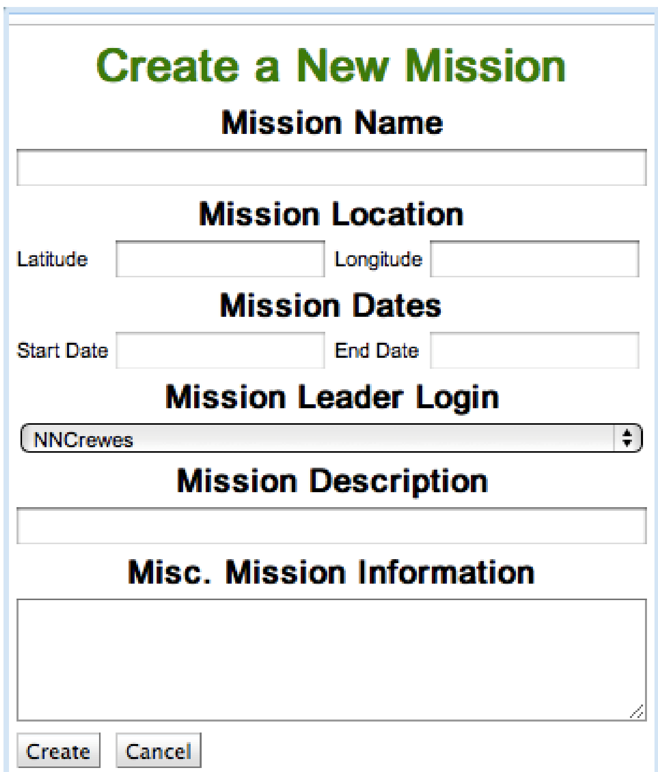


Figure 16. Create New Mission Panel

The next option for the user changes the way the missions are sorted in the Missions Panel. The user can choose to sort missions by order created, mission creator

login, mission start date, and mission end date. The next option the user has from this panel is to display all the missions on the map at once. This will clear the map of any overlays and display all of the current missions on the map. These missions are all clickable to see the mission information as shown in the figures above.

After the mission options are the mission statistics. The mission statistics show the current number of missions (this is the number of missions whose end date is after the current date), the number of finished missions (this is the number of missions whose end date has passed the current date), the total number of missions, and the number of missions created by the user who is logged in.

e. Points of Interest Panel

The Points of Interest Panel is much like the Missions Panel. It displays a list of all the Points of Interests. The list contains an anchor for the Point of Interest name, the time when the Point of Interest was created, the Mission with which the Point of Interest is associated, the creator, and a button that will show the Point of Interest on the map. Since the creator name is actually the creator login, for ease of identification the user name can be moused over to see the creator's full name. Like the mission panel if the Point of Interest's name anchor is clicked then the Point of Interest's description is displayed in the list with a second anchor that says "more…" If this anchor is clicked then a panel is displayed with all the Point of Interest's information. If the creator of the Point of Interest has clicked on the

anchor then he/she has the option to edit or remove the Point of Interest. This flow is demonstrated in Figure 17.

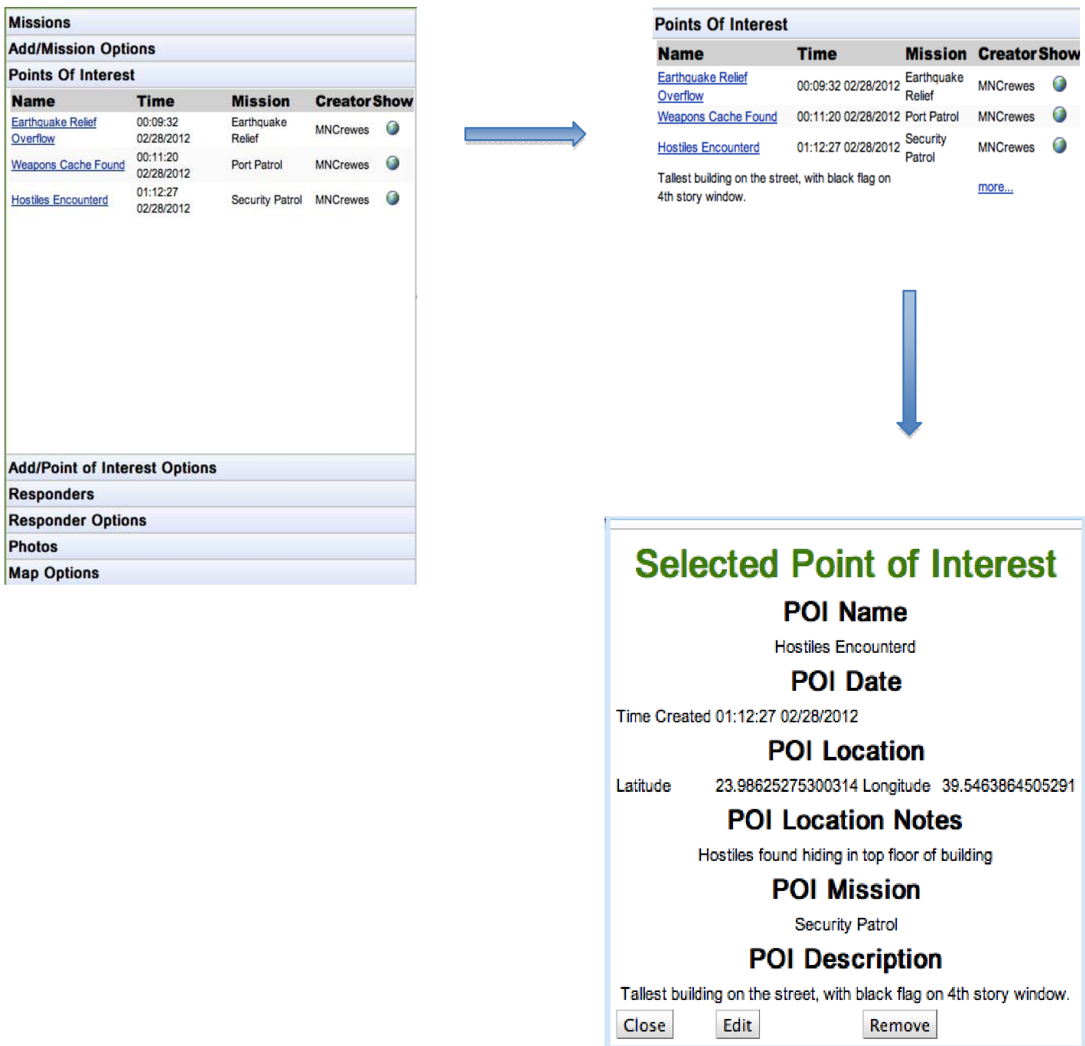


Figure 17. Point of Interest Flow

The globe button allows the user to see the Point of Interest on the map as shown in Figure 18. If this button is pressed then the Point of Interest along with its essential information will be displayed on the map, as

demonstrated in Figure 19. If the bubble on the map is expanded then all of the Point of Interest’s information will be displayed. Within this expanded bubble are links to see corresponding responders, missions, and pictures on the map at the same time. Once these associations are displayed on the map the user can click on them to get their corresponding information, as displayed in Figure 20.

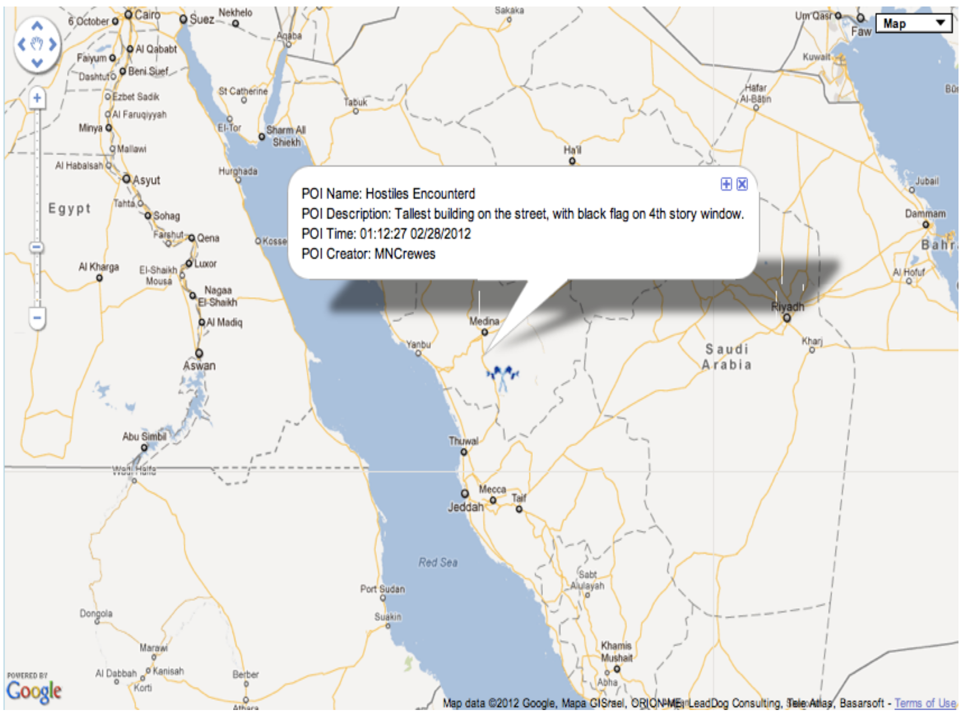


Figure 18. Point of Interest on Map

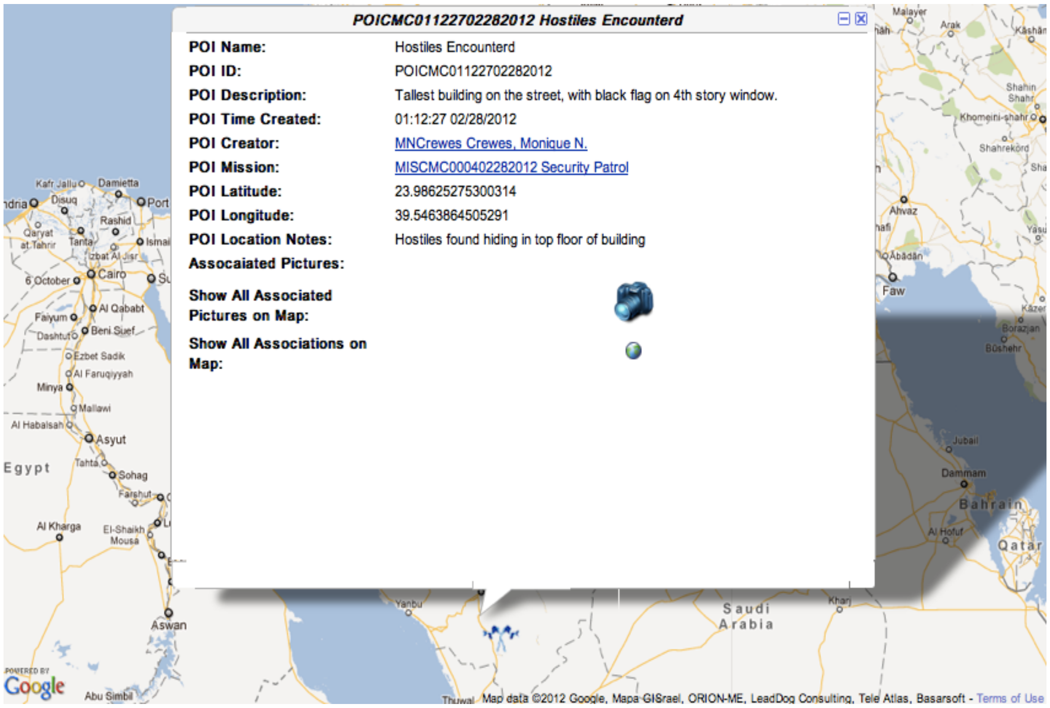


Figure 19. Expanded Point of Interest Information

Figure 20. Point of Interest Associations on Map

f. *Add/Point of Interest Options Panel*

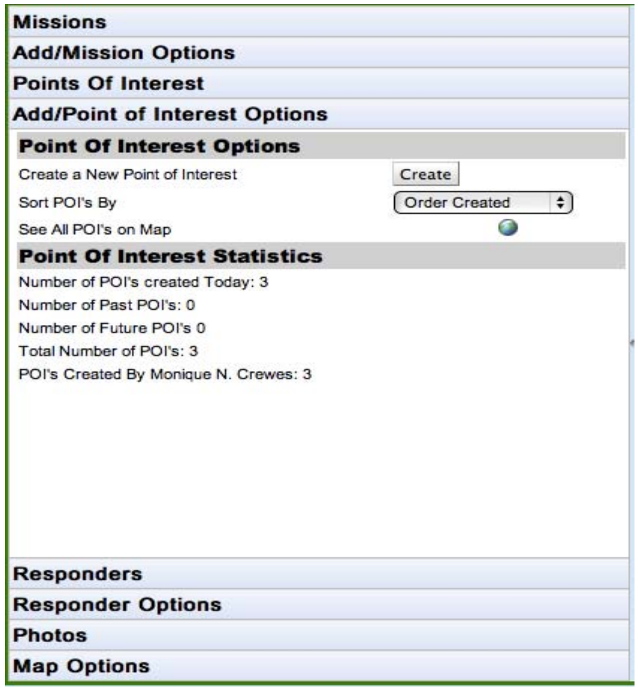


Figure 21. Add/Point of Interest Options Panel

The Add/Point of Interest Options Panel shown in Figure 21 allows the user to add a Point of Interest to the list, access Point of Interest list display options, show all Points of Interests on the map, and show Point of Interest statistics. From this panel the first option is a button for the user to create a Point of Interest. Clicking this button brings up a Point of Interest form displayed in Figure 22 for the user to input Point of Interest information. The Point of Interest must have a name, valid location, and a correlating mission.

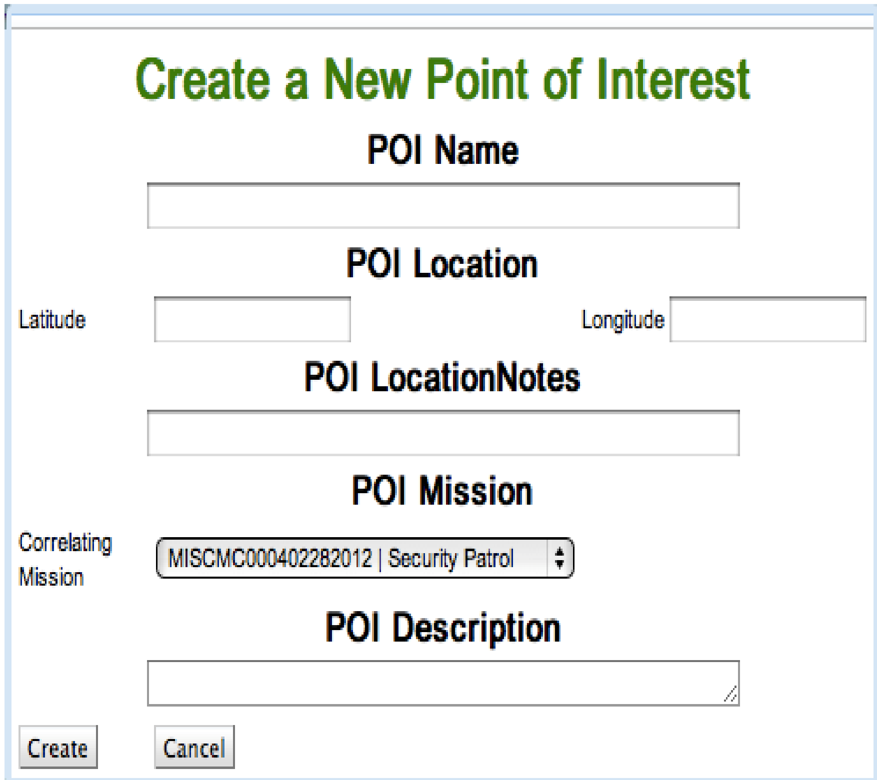


Figure 22. Create New Point of Interest Panel

The next option is for the user to change the way Points of Interests are sorted in the Point of Interest Panel. The user can choose to sort Points of Interests by order created, creator, time of interest, and correlating mission. The next option the user has from this panel is to display all the Points of Interest on the map at once. This will clear the map of any overlays and display all of the current Points of Interest on the map. These Points of Interest are all clickable to view their information as shown in the figures above.

After the Point of Interest options are the Point of Interest statistics. The statistics show the current number of Points of Interests created on the current day, past Points of Interest (this is the number of Points of

Interest whose time stamp is before the current date), the number of future missions (this is the number of Points of Interest whose time stamp is ahead of the current date), the total number of Points of Interests, and the number of Points of Interests created by the user who is logged in.

g. Responders Panel

The Responders Panel contains a list of all the Responders in the SPARCCS system. The list contains the Responder’s login anchor, type, unit, associated mission, and a graphic to indicate if the responder is logged into the system. Since only the Responder’s login is shown if the login is moused over then the user is able to see the Responder’s full name. Like the Mission and Point of Interest Panels, if the login anchor is clicked-on then the Responder’s unit and e-mail is displayed along with an anchor that says “more…” If this anchor is clicked-on then a panel containing all of the Responder’s information is shown. This flow of information is illustrated in Figure 23.

Figure 23. Responder Information Flow

Likewise, there is a globe button associated with each Responder. If clicked, the Responder will display on the map with a bubble containing the Responder's essential information, as shown in Figure 24. If the bubble on the map is expanded then all of the Responder's information will be displayed as displayed in Figure 25. Within this expanded bubble are links to see corresponding missions and pictures on the map at the same time. Once these associations are displayed on the map the user can click on

them to get their corresponding information, as demonstrated in Figure 26.

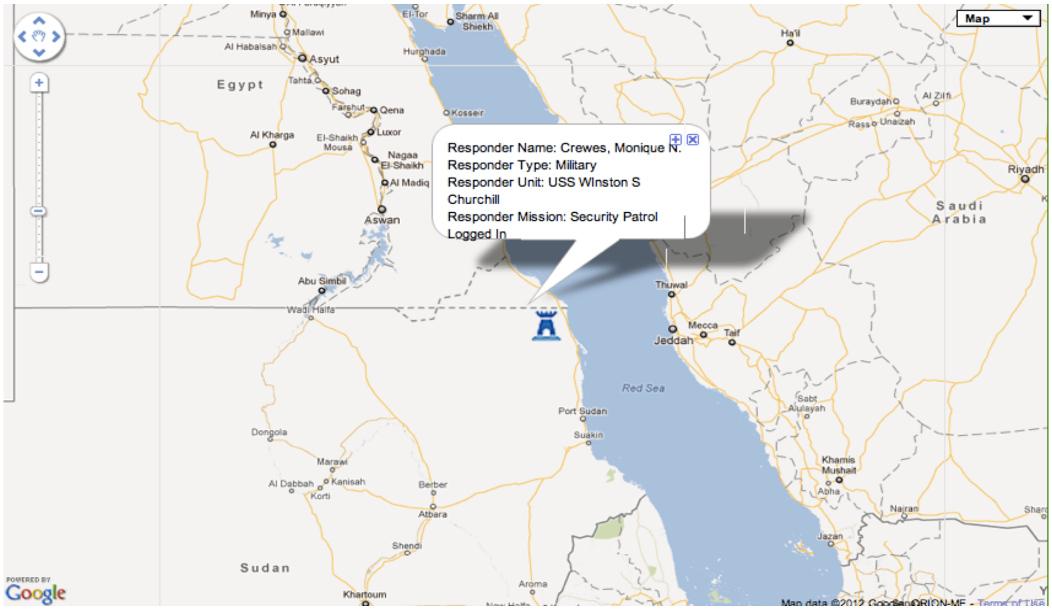


Figure 24. Responder on Map

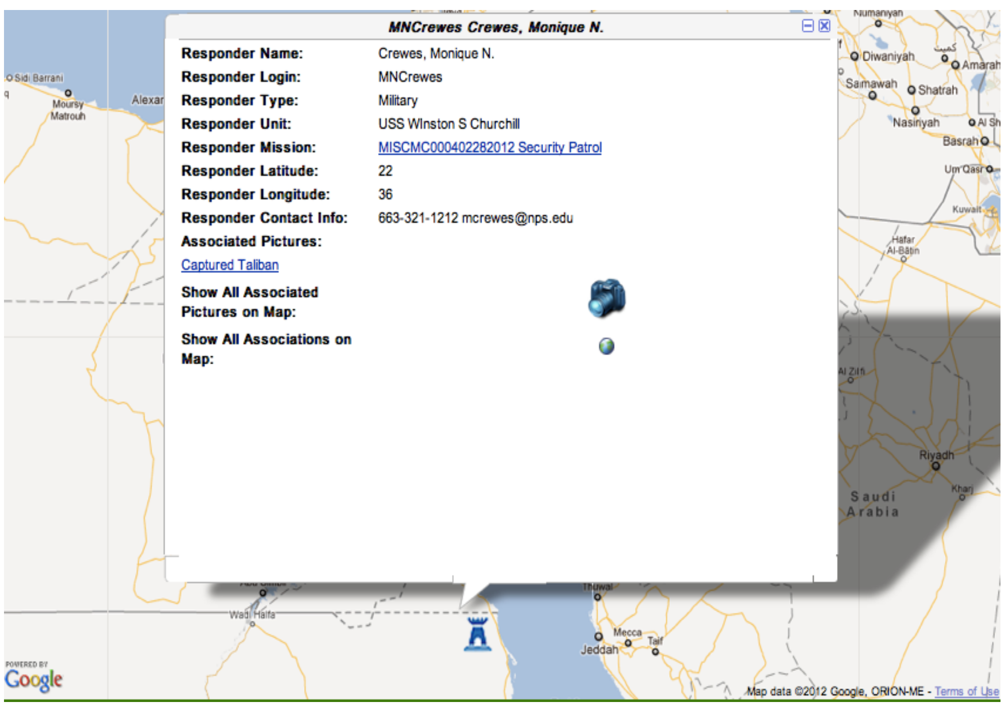


Figure 25. Expanded Responder Information on Map

Figure 26. Responder Associations on Map

h. Responder Options Panel

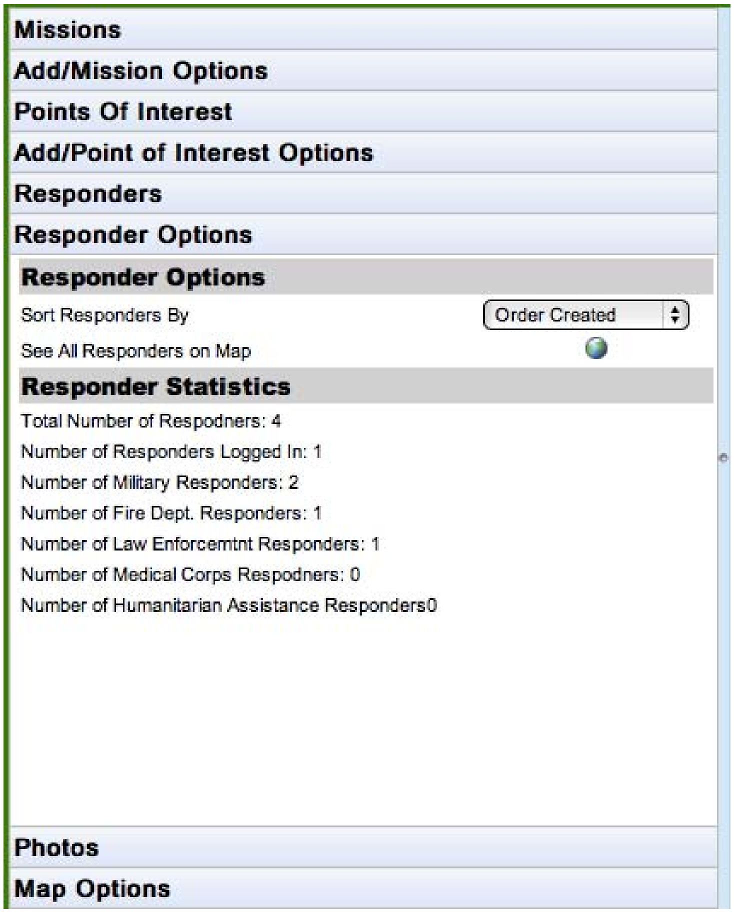


Figure 27. Responder Options Panel

The Responder Options Panel in Figure 27 provides the user the option to sort the Responders list, show all the Responders on the map, and show Responder statistics. From this panel the first option is to sort the Responders. Responders can be sorted by order created, login, type or associated mission. The next option is to display all the Responders on the map at once. This will clear the map of any overlays and display all of the current Responders on the map. All of these Responders are clickable, enabling the user to see their information as shown in the figures above.

i. Photos Panel

The Photos Panel in Figure 28 consists of three main sections: the Missions section, the Point of Interest section, and the Responders section.

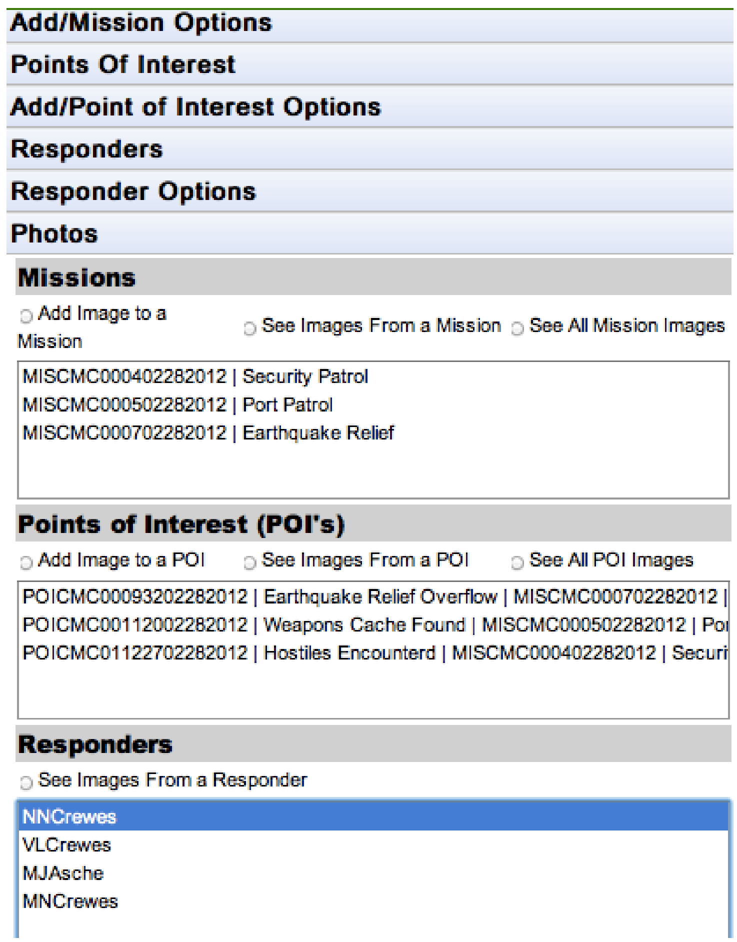


Figure 28. Photos Panel

The Mission section contains three radio buttons and a list box. The radio buttons enable the user to add an Image to a selected Mission from the list box, see an Image from a selected Mission from the list box, or to see all Images from all Missions.

To add an image to a selected mission the user must select a mission from the list box then select the

radio button that says “Add Image to a Mission.” This will bring up a photo form to be filled in by the user. In order to create an Image the user must enter a title, description, valid location, and valid picture type. Once the picture is uploaded the picture will display in the bottom portion of the form with a label containing the Mission name to which the Image was associated. This flow is illustrated in Figure 29.

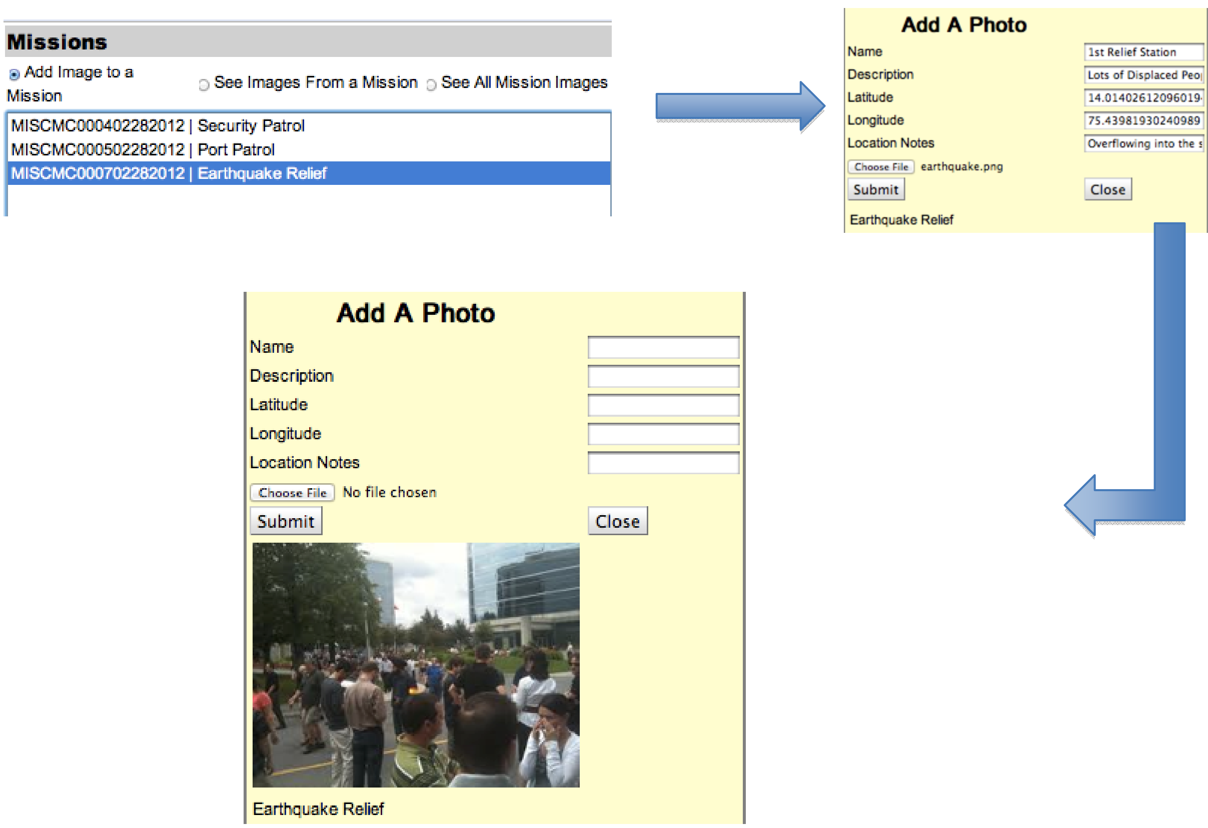


Figure 29. Create Image Flow

To see Images from a mission the process is similar. The user must select a Mission from the list box then select the radio button that says “See Images From a Mission.” This will bring up a photo gallery with

thumbnails of all the images from the mission as shown in Figure 30. If an Image in the photo gallery is moused-over then a small popup will display the Image's description. If the thumbnail of the Image is clicked-on then a second window will display with the Image in its full size with all its corresponding information, as displayed in Figure 31. If the current user created the mission then the user will have the option to delete the Image.

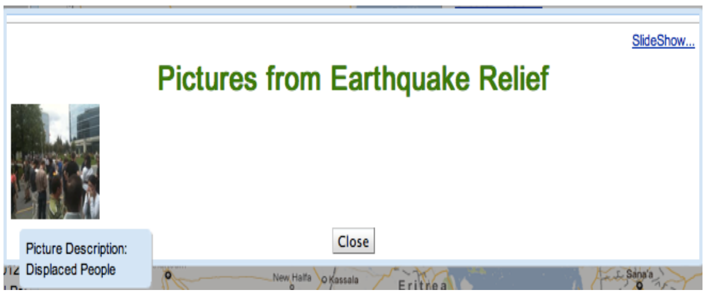


Figure 30. Pictures From Mission

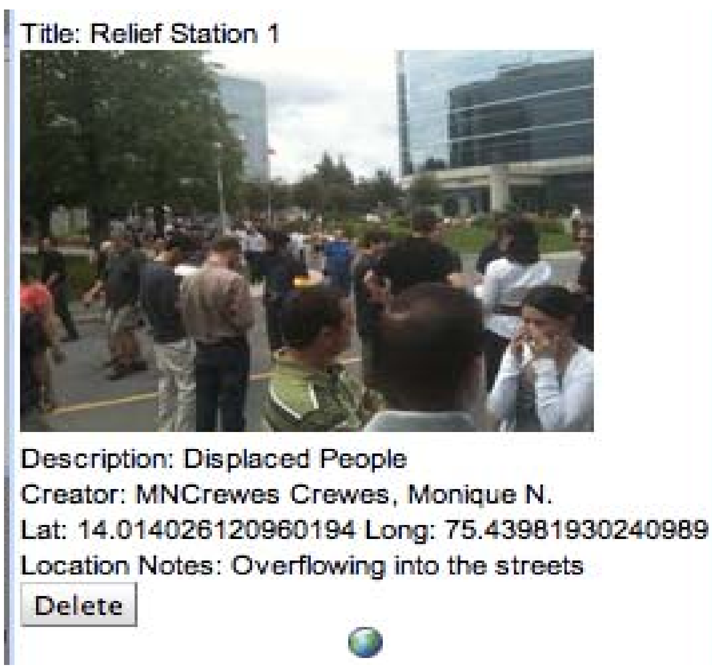


Figure 31. Individual Mission Picture

There is also a globe button on this panel that will allow the user to show the Image's location on the map with a bubble containing all the Photo's essential information as seen in Figure 32. If this bubble is expanded the user will again be able to see all the Image's information, with anchors to see links to the Image's associations on the map as shown in Figure 33.

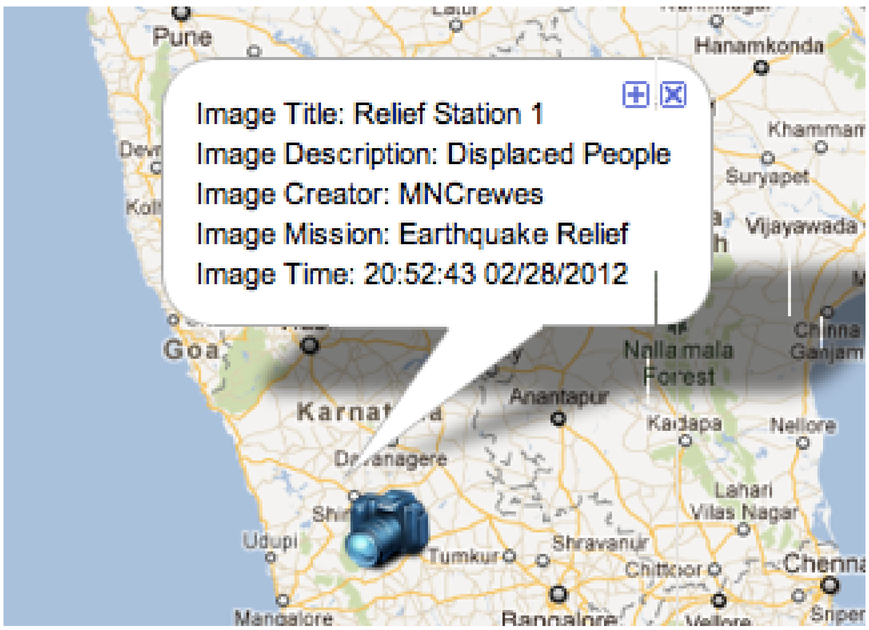


Figure 32. Picture on Map

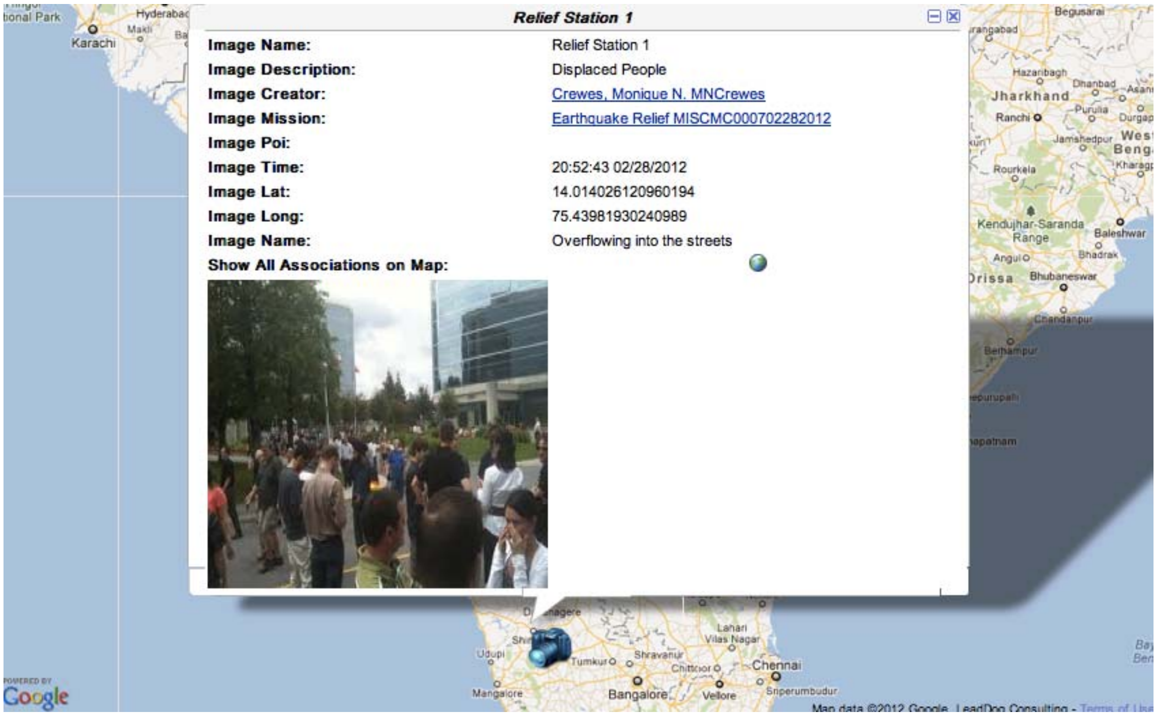


Figure 33. Expanded Image Information

To see all Images from all Missions the user simply clicks on the radio button “See all Mission Images.” This will bring up a photo gallery containing thumbnails of all the Images associated with all missions as seen in Figure 34. The same actions from within the photo gallery apply.

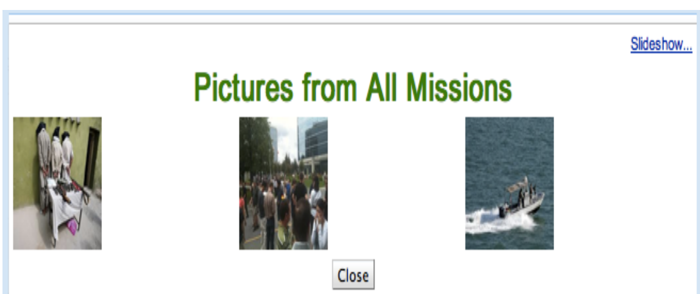


Figure 34. Photo Gallery of All Mission Pictures

For all photo galleries there is an anchor for a slideshow in the top right corner. This will allow the user

to see the Images from the photo gallery in a larger slideshow with all of the Images' information displayed at once as demonstrated in Figure 35.

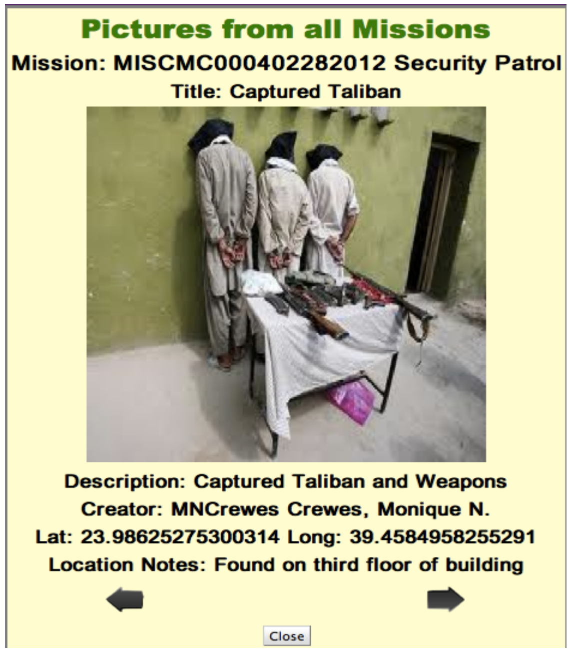


Figure 35. Photo Slideshow

The Point of Interest Photo section operates in the exact same way as the Mission section. The user is able to add Images to Points of Interest, see Images from a selected Point of Interest, and see all Images associated with a Point of Interest in the same manner as described above. The Responder section also operates in the same manner, with the exception that the user cannot add photos in this section. The user can only view Images from a selected Responder by selecting the Responder's login from the list box.

j. Map Options Panel

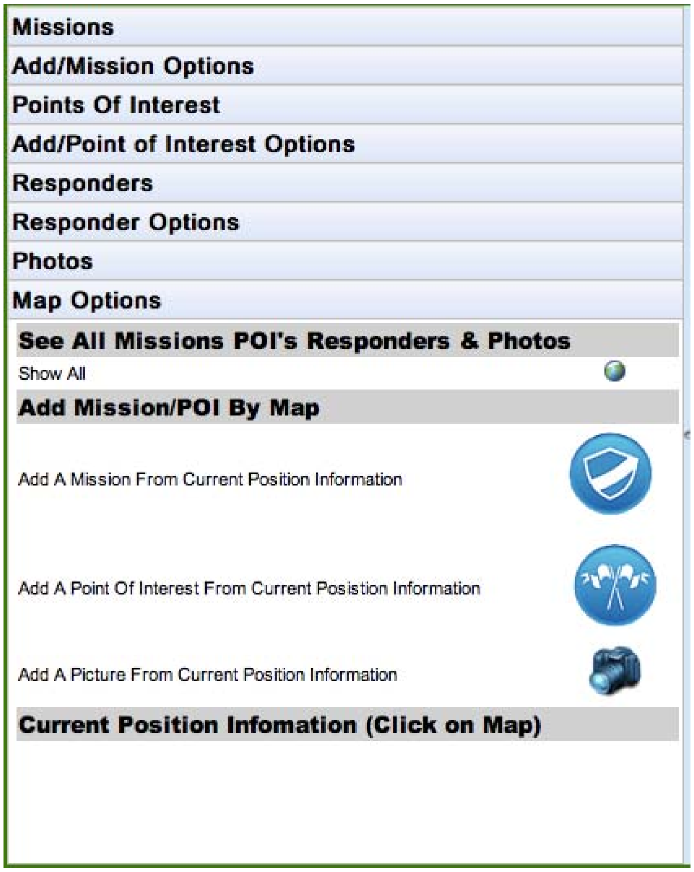


Figure 36. Map Options Panel

The Map Options Panel, shown in Figure 36, is the final panel available for selection. This panel contains options for operating within the map portion of the application. The panel is split into three sections. The first Section allows the user to see all Missions, Points of Interest, Responders and Images on the map at one time in the same manner as when the application opens. The second portion of the panel contains three buttons that allow the user to create a Mission, Point of Interest or Image from a position the user has selected on the map. If any one of these buttons is clicked the corresponding

creation form, as shown earlier, will be displayed with the latitude and longitude boxes already filled in with the coordinates of the position the user clicked on the map. The third portion of the screen, as shown in Figure 37, displays a detailed geo-location as well as the latitude and longitude from a position selected on the map. Also if the user would like to clear the map of all overlays the user can click the clear button at the bottom portion of this section.

Figure 37. Current Position Information

k. Map Pane

The Map Pane is consists of the right hand potion of the application taken up by a Google Map. There is a division between the map and the navigation panel on the left that can be adjusted to see more or less of the map as the user requires. The right hand navigation pane will adjust to these increases and decreases in width to remain readable. The user may have the map take up the whole screen. The map contains large controls both on the top right and the top left. The controls on the top right enable to user to zoom in and out on a current position as well as drag the map to a particular location. The controls on the top right enable the user to select the type of map

to use. There are four options: the standard map containing roads and landmarks, the satellite map produced from satellite imagery, the terrain map showing basic topographic features, and the hybrid view overlaying street names on the satellite view. If the map is clicked in any view at any time the map will provide an overlay at the location clicked with latitude and longitude and, if available, further geocoded information of the position selected. In addition to the information on the positions, anchors are provided that allow the user to create Missions, Points of Interest, or Images, as displayed in Figure 38. The same respective forms will show up as shown in the Figures above, the only difference being that the latitude and longitude text fields will be filled in from the position clicked on the map.

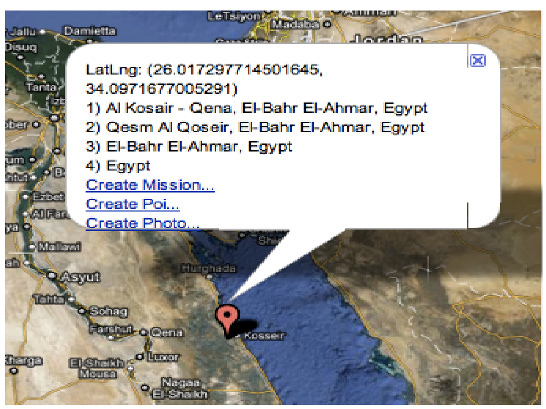


Figure 38. Position on Map

2. Android Client

a. Login Screen

Like the cloud application, the Android application begins on the login screen shown in Figure 39. To prevent the operation of the application in non-logged-

in mode, the login screen cannot be closed without either logging in, or exiting the program.

The Login screen contains two fields and three buttons. One field is for the user name and the other is for the password. The three buttons are marked “login,” “create account,” and “cancel.” The login button sends the username and password to the cloud server. A successful combination of the two prompts the cloud server to send back all of the user information associated with the login name. The cancel button simply exits the program. The create account button opens another form, as shown in Figure 40. The form contains fields for first, middle, and last name, unit, phone number, e-mail, and a spinner for responder type. The create user form has two buttons on the bottom, a save button and an exit. Both buttons cause a return to the previous screen, but pressing the save button sends all the entered data to the server. If an account is successfully created the user’s new username will be

visible in the username field of the login screen. The username consists of the user's first and middle initial, their last name and a number if a similar username alread exists. Currently this process is done autonomously at the cloud server but future versions of SPARCCS should include some sort of authentication to prevent unauthorized access to the system.

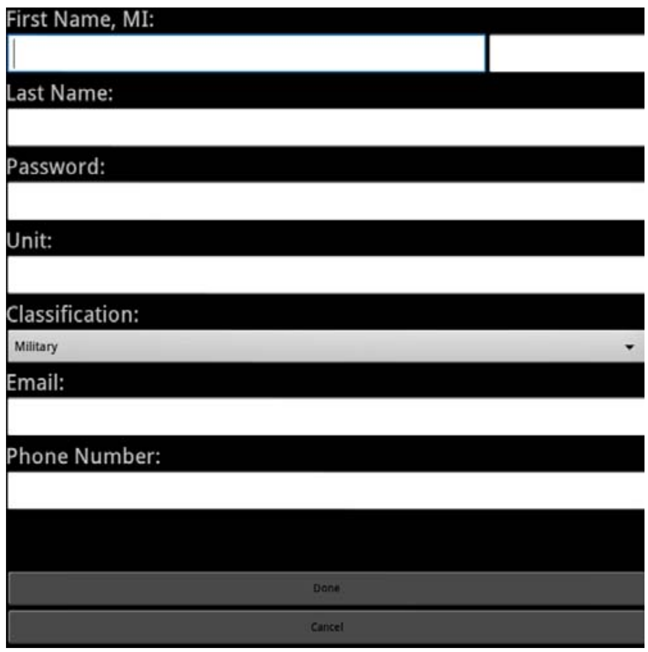


Figure 39. Create Account Form

Upon logging in and receiving the user data back from the server, the application checks the user information to make sure the user is part of a mission and that the user's mission still exists. If so, the user is taken to the main screen. If not, a dialog box opens informing the user that they are not part of a mission shown in Figure 41. The dialog box contains two buttons, one allowing the user to create a new mission, and one allowing the user to join an existing mission.

Figure 40. No Mission Dialog

b. Main Screen

Once the user is logged-in and it is verified that they are part of a mission, they are taken to the main interaction screen, shown in Figure 42. The screen is simply a Google Map view with a radio button in the top right corner for toggling between map and satellite mode. Users have the ability to change the map center by placing one finger anywhere on the map and dragging it in any direction. Users are able to zoom in on an area of the map by placing two fingers on the map and dragging them apart from one another. Conversely, placing two fingers on the map and dragging them together zooms out.

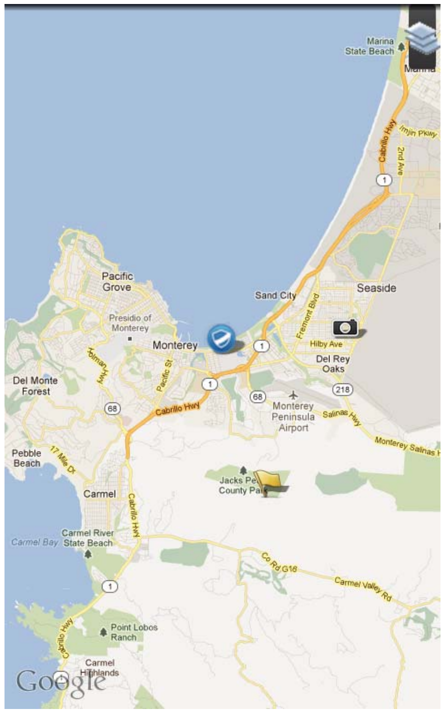


Figure 41. Main Interaction Screen

c. *Menu Button*

Unlike the web application, screen space is very limited on the mobile application. For this reason, apart from the map/satellite radio button, there are no specific function buttons on the screen. Pressing the Android menu button accesses all other functionality. Pressing this opens a menu at the bottom of the screen with four options: users, points of interest, missions, and images. This options menu is displayed in Figure 43.

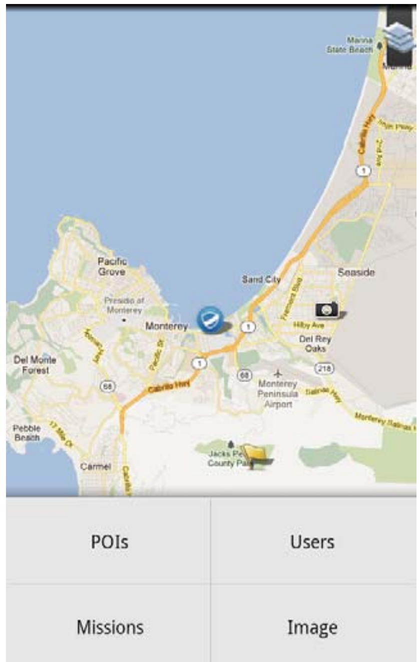


Figure 42. Main Screen With Options Menu

Pressing any of these four buttons will open a submenu. The submenu items available after pressing the responder button, shown in Figure 44, are “My Info,” which opens a responder form, and “List,” which opens a list view of all users. After pressing the point of interest or the mission button, shown in Figure 45 and 46, the options are “New,” which opens either a new point of interest or mission form, and “List,” which opens a new list view. The camera button sub items, shown in Figure 47, are “From Gallery,” which takes users to their picture gallery, and “Take,” which opens the device’s camera.

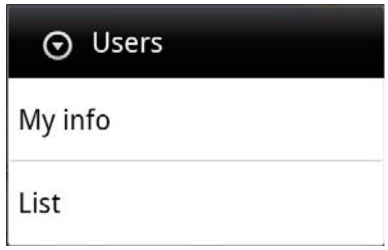


Figure 43. User Submenu Options

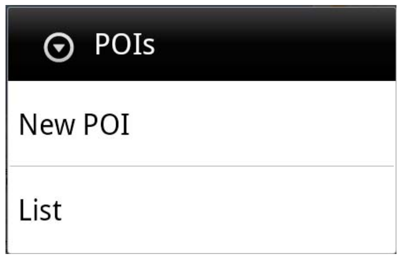


Figure 44. Point of Interest Submenu Options

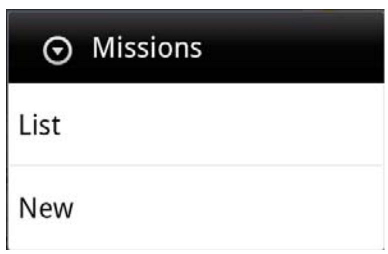


Figure 45. Mission Submenu Options

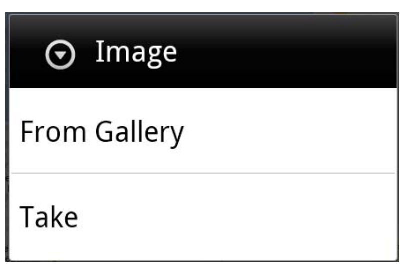


Figure 46. Image Submenu Options

d. Missions

Missions can be created, joined, viewed or edited by a user; but only a system administrator can delete a

mission. Creating a mission can be done one of two ways: either by pressing the create mission button on the dialog that opens when a user without a mission logs-on, or by pressing the menu button, followed by the mission menu button, and then the new button.

The new mission form, shown in Figure 48, has three fields: the mission title, the mission description, and any miscellaneous mission information. In addition, there are buttons, labeled “Start date” and “End date.” Pressing either of these buttons opens a date picker dialog that allows users to select a date.

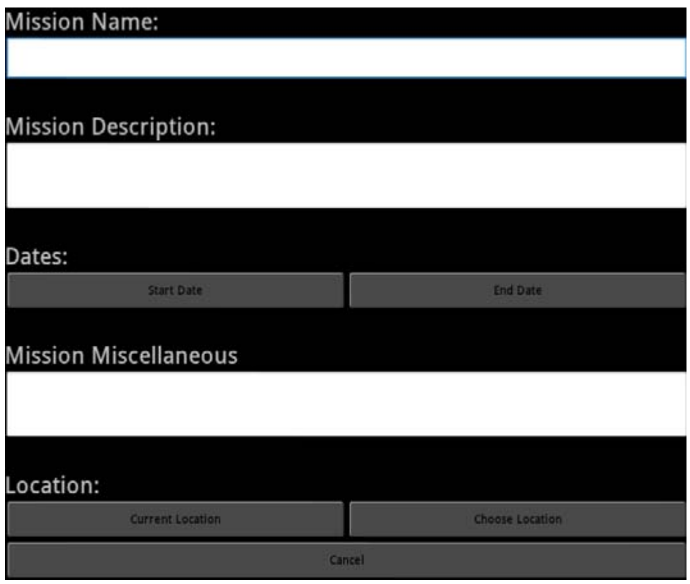


Figure 47. Mission Form

At the bottom of the form, under the label “Location:,” there are three additional buttons labeled “Current Location,” “Choose Location,” and “Cancel.” Pushing any of these buttons returns the user to the initial screen. If the “Current Location” button is pressed the system places a blue shield marker at the current location of the device. It also opens a menu asking the

user if they would like to save the mission, choose a new location, or cancel. If the user had pressed the “Choose Location” button the system starts an “on-tap-listener” and will place a marker at the location of the next tap on the map. Following the marker placement, the menu opens asking if the user would like to save the mission, move the mission location, or cancel mission creation. This menu can be seen in Figure 49. Had the user pressed the “Cancel” button from the mission form all mission information would be discarded.

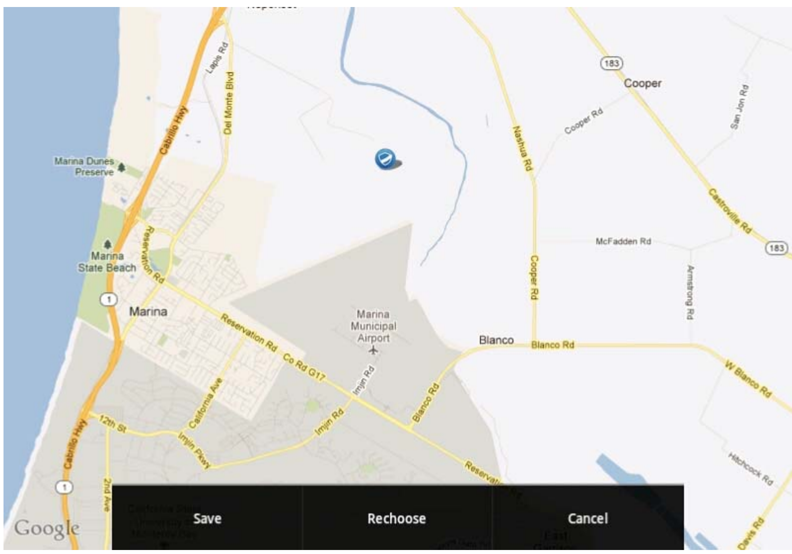


Figure 48. Mission Save Options

Once a mission is created it can be viewed in two ways. The first, and usually the simplest, is to tap on the mission icon from the map. Mission information can also be viewed by pressing the menu button, followed by the mission menu item, followed by the list submenu item and finding the mission title. A screen shot of this is shown in Figure 50. Performing either of these actions will reopen the mission form with the mission’s information filled-in. The

mission form has two additional entities: a spinner containing the mission members with the mission leader selected, and a field indicating the mission creator.

If the user viewing the mission is the mission leader the user has the ability to edit the mission description, dates, or miscellaneous data fields, or switch the mission leader to another member in the mission. In this case, there are two buttons on the bottom, one to save changes and one to cancel any changes that may have been made. If the user is not the mission leader but a member of the mission, there is only one button at the bottom to take the user back to the initial screen. Finally, if the user viewing the mission is not a member of the mission there are three buttons at the bottom. The first, “Show Entities On Map,” will result in the system displaying all associated points of interest and image markers on the map. The second button, “Join Mission,” switches the user’s current mission to the one being viewed. The final button, “Back,” returns the user to the initial screen with no changes being made.

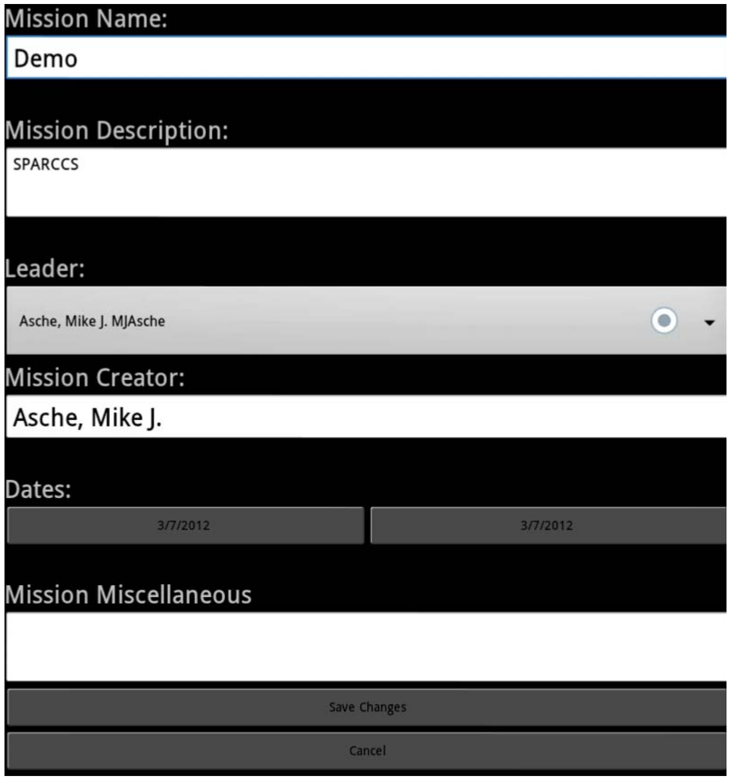


Figure 49. View Mission Form

e. Points of Interest

Pressing the menu button, followed by the POI button, and the new button creates Points of Interest. Once this has been done the application calls a new activity containing the Point of Interest form, shown in Figure 51. The Point of Interest form has three fields: POI Title, POI Description, and Location Notes. On the bottom of the form, under the text "location," there are three buttons: "Current," "Choose," and "Cancel." Pressing "Cancel" discards the point of interest and returns to the main screen. Pressing "Current" retrieves the user's location and returns to the main screen where the map is centered on a flag placed at the device's current location. Pressing the "Choose" button also returns to the main screen,

however, the next time the screen is tapped a flag will be dropped at the location of the tap.

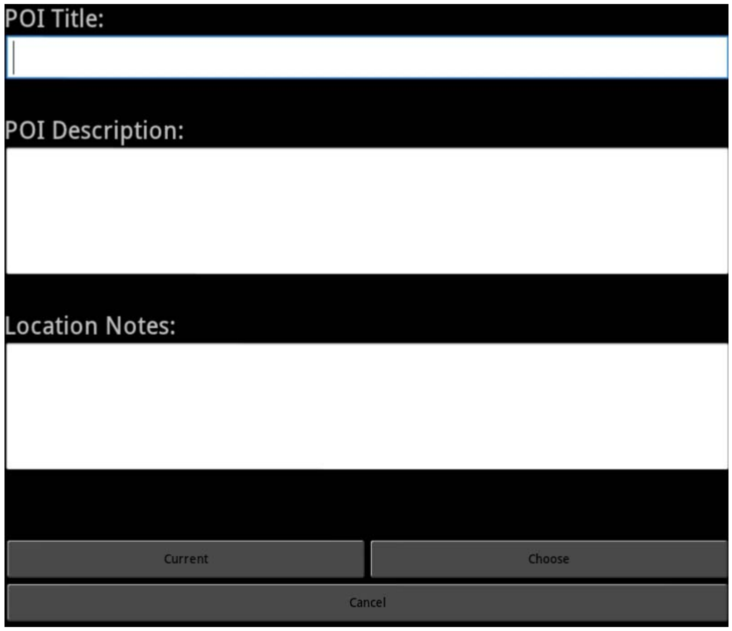


Figure 50. Point of Interest Form

Once a flag is dropped, either from pressing current location or tapping the screen, a new menu opens at the bottom of the screen. The menu, shown in Figure 52, give’s users the opportunity to save the POI, move its location, or cancel the action. Pressing “save” adds the point of interest information into the phone’s local database, as well as adding it to a queue of information to be sent the next time the phone connects with the server.

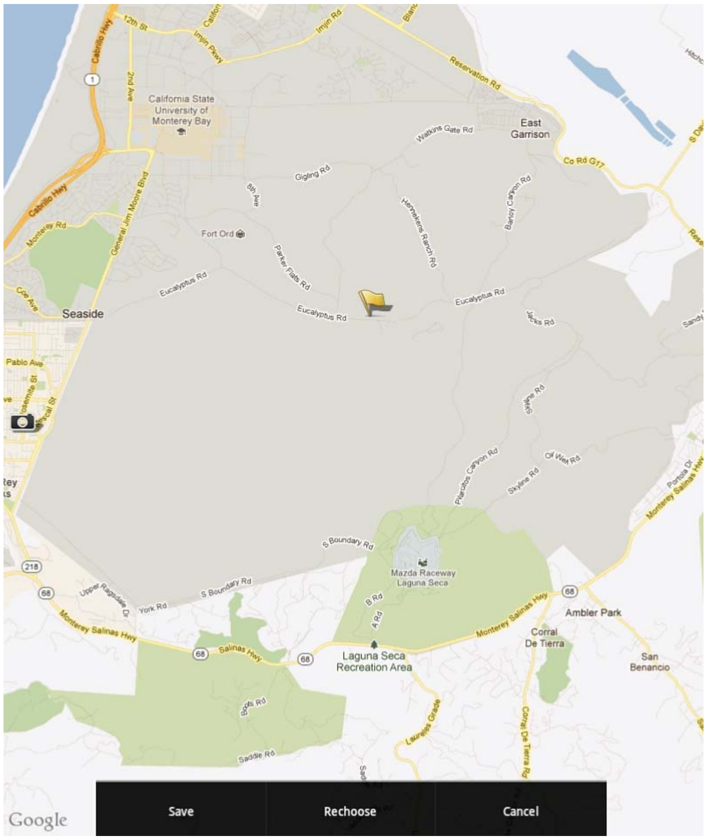


Figure 51. Point of Interest Save Options

The Point of Interest information form, shown in Figure 53, can be called in two ways. The first is by finding the point of interest on the map and clicking on its icon. The second is by finding the point of interest in the list view. Either way, the device presents the point of interest form filled in with the point of interest's information. The form additionally has fields for time created, creator, and associated mission. There is also a list view containing the name of any image associated with the point of interest. Clicking on any of these items will open an image form.

If the user viewing the point of interest is the user that created it he will have the ability to edit the

point of interest’s description and location notes or delete the point altogether.

Figure 52. View POI Form

f. Images

Images can be entered into the system in two way: either by capturing an image with the device’s native camera, as seen in Figure 54, or by selecting a previously captured image from the photo gallery, shown in Figure 55. This option is presented to users by pressing the menu button followed by the image menu item. When an image is captured it is saved as a .tmp file in the applications folder. The path to the image is saved in ImageClient class and will be used to access the picture until it is uploaded to the server. Once the image is successfully uploaded it

is assigned a URL for accessing the picture and the .tmp file is deleted.

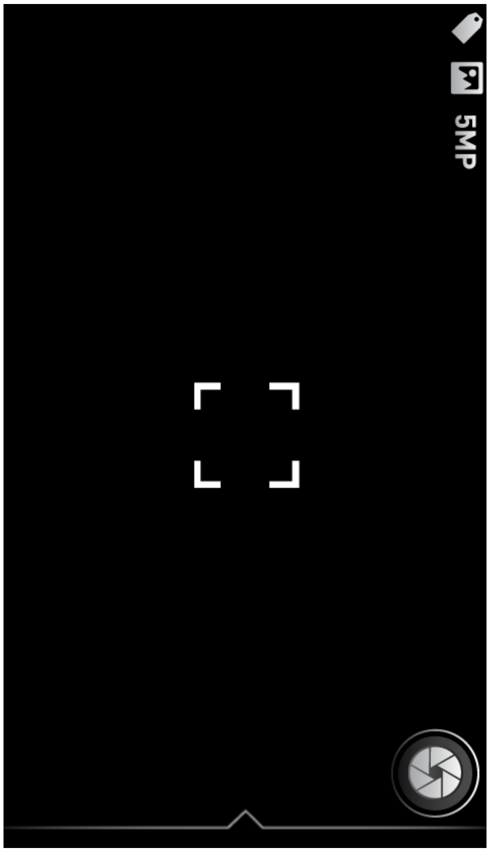


Figure 53. Android Native Camera

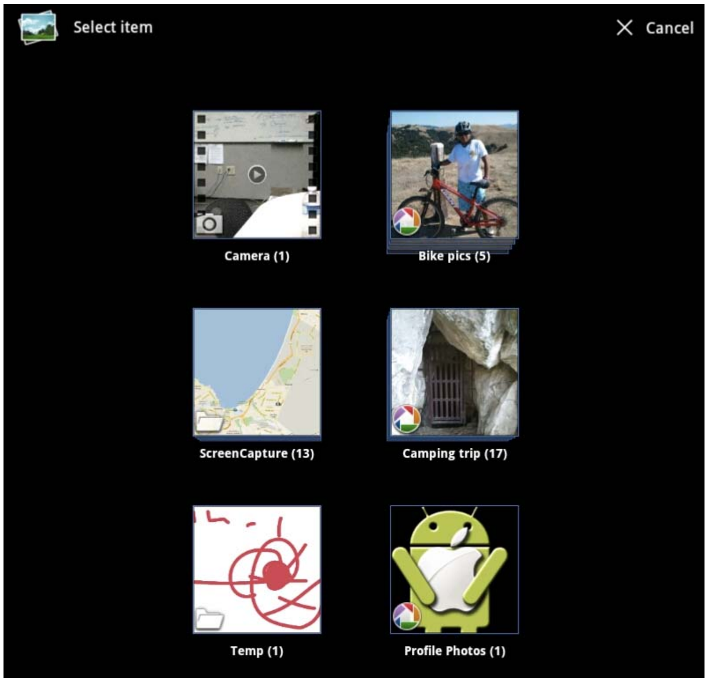


Figure 54. Android Photo Gallery

Immediately following either capturing a picture or choosing one from the gallery, the image form is opened, shown in Figure 56. The image form displays the image at the top of the form, and beneath it there are forms for entering the image name, description, and location notes. There are, additionally, three buttons for indicating the image location. Users have the choice of placing the image at their current location, choosing a location on the map, or associating the image with a point of interest. The first two options will place a camera icon on the map at the location designated. The third option does not place an icon on the map. Either of the first two options will open an image “save option” menu giving the user the choice to save the image, move the image’s location, or cancel the image creation. This menu option is shown in Figure 55.

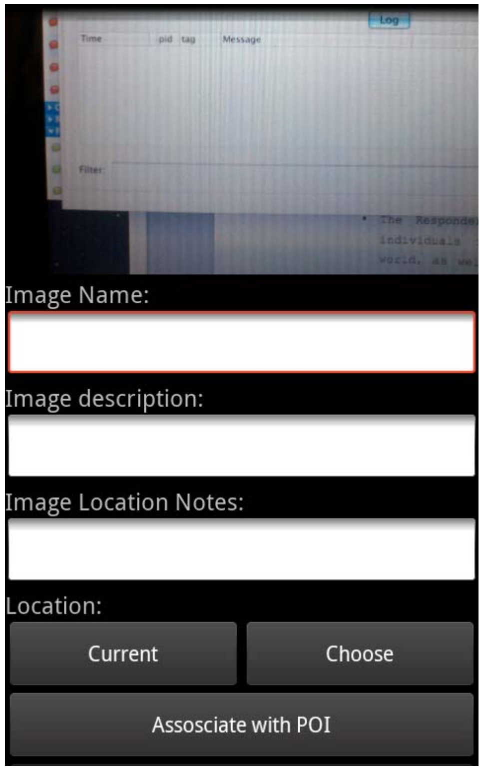


Figure 55. Image Form

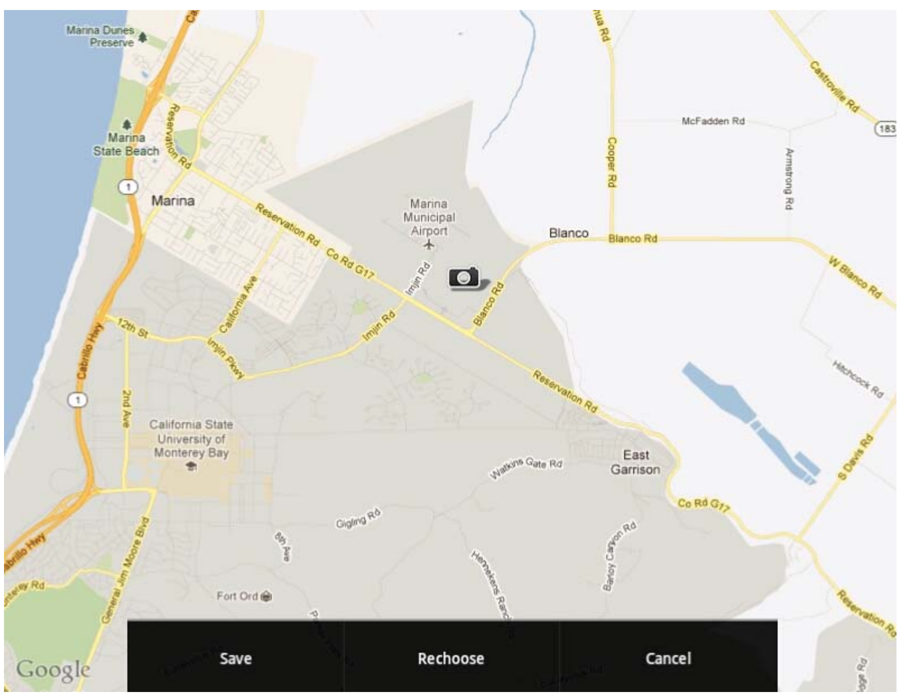


Figure 56. Image Save Options

Images can be viewed by either tapping on the image’s icon, or if associated with a point of interest, by opening the point of interest form and clicking on the image title under the label “Associated Images.” Viewing an image reopens the image form, shown in Figure 58, with additional fields for the image’s associated mission, point of interest, and the image creator. If the user viewing the image is the image creator she has the ability to edit the image description and location notes or to delete the image. Images are not stored locally on the device for memory constraint reasons. Instead, each image is assigned a URL when it is uploaded to the server and the local .tmp file is deleted, however locally generated images remain in the user’s gallery. When a user opens an image form for viewing an image the system downloads the image from the server then displays it on the form.

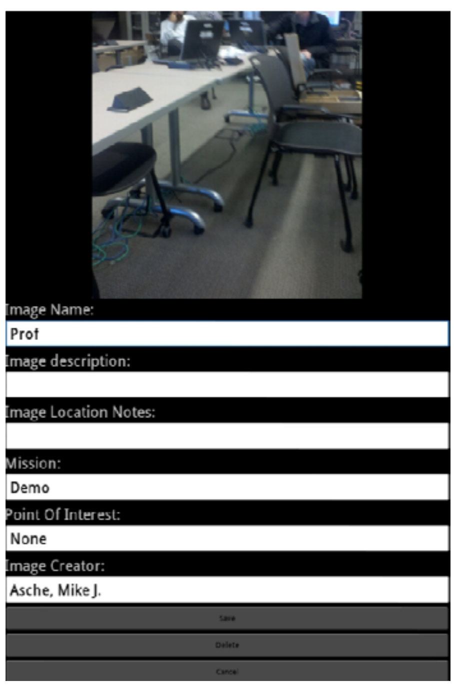


Figure 57. View Image Form

g. Users

User information can be viewed by pressing the menu button, followed by the user menu item, followed by the list sub-item then selecting a user. Once this is done, the user form is opened filled-in with the user's information. The form has fields for first, middle, last, and user name; as well as unit, current mission, classification (military, fire protection, law enforcement, medical corps, or humanitarian assistance), e-mail, and phone number. Clicking the done button at the bottom of the form returns the user to the list view.

By pressing the menu button, followed by the user menu item, and the "my info" submenu item, a user can view and edit their own information, as shown in Figure 59. Editable fields include the user's unit, e-mail, and phone number.

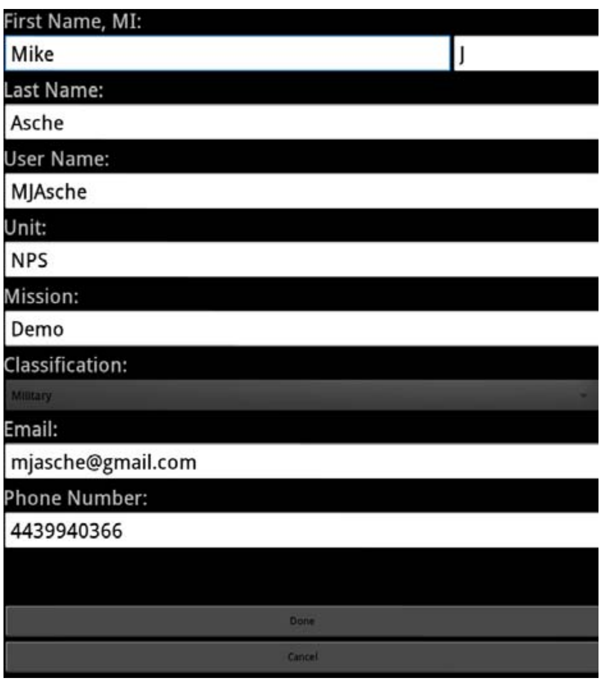


Figure 58. View User Form

h. List Views

List views are available for Responders, Points of Interest, and Missions. Pressing the menu button, followed by the desired menu option, and then the “list” submenu item, accesses these lists views. Initially, each list view starts by sorting the items alphabetically by name. However, each list view has different options for sorting items. The options for sorting missions, shown in Figure 60, include by name, by start date, by end date, and by leader. In the “point of interest” list view, shown in Figure 61, options include by name, by mission, by creator and by creation time. Finally, the options for sorting users, shown in Figure 62, are by name, by mission, by unit, and by type.

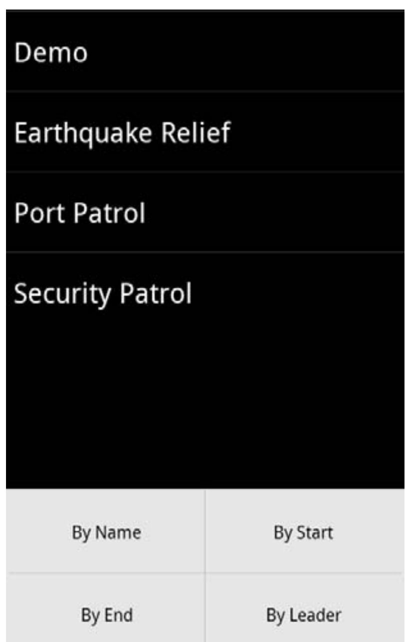


Figure 59. Mission List View

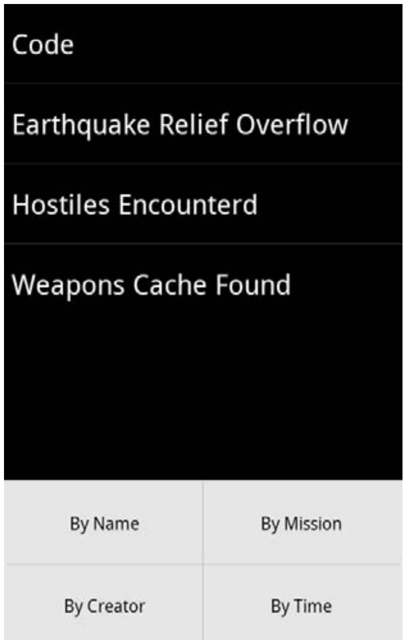


Figure 60. Point of Interest List View

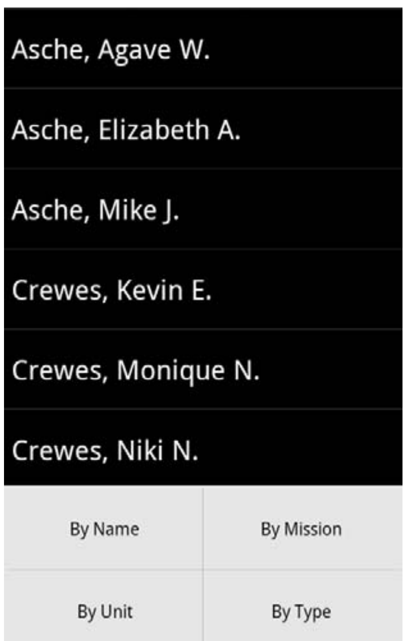


Figure 61. User List View

i. SQLite Database

All information, aside from images, is stored locally on the phone’s native database. When the application is run for the first time on any device, the system creates a database with 15 tables. The tables are as follows:

- responder – stores the list of all responders in the system.
- pointOfInterest – Stores the list of all points of interest in the system.
- sendPointOfInterest – Stores the list of locally created points of interest that have yet to be uploaded to the Server.
- editPointOfInterest – Stores the list of locally edited points of interest that have yet to be modified in the server.
- mission – Stores the list of all missions in the system.
- sendMission – Stores the list of locally created missions that have yet to be uploaded to the server.
- editMission – Stores the list of locally edited missions that have yet to be modified in the server.
- image – Stores the list of all images in the system.

- sendImage – Stores the list of all locally created images that have yet to be uploaded to the server.
- editImage – Stores the list of all locally edited images that have yet to be modified in the server.
- myInfo – The same setup as the responder table, but only holds the information for the current user.
- deletePointOfInterest – Stores the point of interest identifiers for locally deleted points of interest that have yet to be deleted from the server.
- deleteImage – Stores the image identifiers and associated blob keys (URLs) of locally deleted images that have yet to be deleted from the server.
- viewMission – Stores a list of mission identifiers whose entities the user would like displayed on the map.
- lastLogin – Stores the username of the last user logged-in so they don't have to reenter their username each time they log in.

j. Syncing

Upon successful login to the system, a synchronization thread is created. The constructor of the thread does an initial pull from the server retrieving all users, missions, images, and points of interest and

inserting them into the database. Once this is done, a run function is called that starts a loop that will continue until the program is exited. The first operation of this loop is to update the responder information. While currently this would only be required if the user information has changed, having it done every time will be helpful when blue force tracking is integrated into the system.

After updating the user information, the loop pulls the list of missions to be sent from the database. Each time a mission is successfully uploaded to the server it is removed from the sendMission table of the database. If there are no new missions to be sent this section is skipped. This process is then repeated for Points of Interest to be sent, Images to be sent, Points of Interest to be deleted, Images to be deleted, Points of Interest to be edited, Images to be edited, and Missions to be edited.

Following the upload of all new information the system does another pull for all Responders, Missions, Images, and Points of Interest. Their corresponding tables are cleared in the database and the new information is added. To prevent new entities from disappearing in the unlikely event that they were added between upload and download, the system checks the send tables and adds their information to the regular tables as well.

Once the sync is complete and the new information is entered into the database, the thread sends a message to the application, via a message handler. The message simply indicates that synchronization has been completed, prompting the application to update the map overlays.

Finally, the thread enters a sleep period. For testing purposes this sleep has been set to thirty seconds. Through field test, an appropriate sleep period should be determined to maximize battery life while minimizing lag time.

C. CONCLUSIONS

In conclusion, the web based cloud application and the Android application take relevant information and present it in an easy to use visual manner. The user can glance at a map or get a more detailed list view of any pertinent information for the operational picture. In the end, the commander and his staff at the headquarters station needs only to bring up a web browser to have a common operational picture for all the operations in their command. For soldiers operating in the field, they only need an Android phone to maintain situational awareness.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

The Smartphone Assisted Readiness, Command, and Control System provides an excellent tool for creating a common operational picture. Users are not only able to constantly view a map of their operating environment, but they are given the ability to modify and share the map, thereby increasing the situational awareness of all parties involved. This thesis shows that not only is it possible to utilize cloud computing in conjunction with smartphone technology (Android-based smartphones in our case), but that it is extremely useful.

By moving the server to the cloud, we have dramatically increased the availability of the server, as it is operated and managed as a dedicated service for a multitude of services. In addition, we have made the application easily accessible, as the only requirement for access is an Internet connection. By utilizing the cloud we have dramatically reduced the cost of operation as it pools computing resources. Finally, we believe deployment of the application will be faster as there is no need to build out the network infrastructure for the server.

By developing the corresponding mobile application, we have extended the common operational picture to the field. Field operatives no longer have to rely on command centers to relay pertinent information to them as they now have the same information the command center has. Likewise, command centers do not need to worry about receiving reports, mapping points of interest and entity locations, or

relaying that information to all other entities. Furthermore, we have taken advantage of a COTS product and its capabilities. Using a COTS product increases the speed of deployment of the application as many responders will already have the devices and just require the application. It also further cuts cost, as a specific device does not need to be designed for the sole purpose of running SPARCCS.

While SPARCCS is not a complete, fully robust system yet, we have provided the framework for a very useful tool for enhancing situational awareness in disaster relief and military scenarios. We built the server, mobile application, and headquarters application, enhanced them with useful functionality, and specified the communication protocols. With some additional functionality and proper field-testing, the SPARCCS program will be ready for deployment.

B. FUTURE WORK

This thesis provides the foundation for building a robust common operational picture tool. While our goal was to make the system as operational as possible, time and manpower limitations prevented us from providing a fully functional system. As such, we provided a list of recommendations to make SPARCCS a field-worthy application.

- Blue Force Tracking—The current SPARCC system utilizes GPS for placement of missions, points of interest, and images at the current location of the user’s device. GPS functionality should be extended to report the location of the device

every time synchronization between the device and the cloud service occurs. Furthermore, the reported location should be forwarded to all other devices and the locations of users should be displayed on the map.

- Phone-to-phone syncing—In the unlikely event that the cloud service goes down, or if devices lose the ability to communicate with the cloud, users should be able to continue utilizing the system by communicating device-to-device rather than through the service. In order for this to be possible, the SPARCCS application needs increased functionality that would enable a particular device to act as an HTTP server. It is our recommendation that each mission leader's device act the HTTP server. All other members of the mission will perform their syncs with the mission leader's device, and when possible, the mission leader's device will sync all mission information with the cloud service.
- Improved mapping capability—Currently, the SPARCCS system can only display points of interests as a single point on the map. This functionality should be extended to accommodate for areas of interest. Future versions of SPARCCS should incorporate polygons, circles, and white boarding in order to highlight larger regions on the map. This functionality will be useful for pointing areas of interest such as fires, floods, radioactive fallout, etc.

- User communications—Current user information includes the user’s phone number and e-mail. With this information users have the ability to contact each other using the device’s native e-mail and cellular capabilities. Future versions of SPARCCS should make communication between users easier by incorporating group and individual chat capabilities as well as push-to-talk capabilities using either Voice Over IP or existing cellular protocols.
- Commands—an additional form of communication between users should be commands. Often times it will be crucial for command centers to issue commands to mission leaders, and mission leaders to mission members. When a command is received it should demand the system’s focus. The system’s focus should remain on the command until the receiving user either acknowledges or declines it. Declining a command should require that the user provide a reason for being unable to comply.
- Routes—While SPARCCS mapping capabilities are extremely useful for location-awareness; it is often helpful to see the route to an intended destination. Future versions of SPARCCS should take advantage of Google Map routing capabilities to enable users to retrieve and display routes on the map. This functionality could further extend the command functionality by enabling users to send routes as a part of a command that includes a destination.

- Video—While the ability to exchange pictures is a key functionality of SPARCCS, users should not be limited to only still images. Most current smartphones have video capabilities. As such, SPARCCS should allow users to capture and share videos.
- Picture editing—In order to further extend the usefulness of sharing images, users should have the ability to edit the pictures they capture. The merit of this functionality is that it would allow users to highlight certain parts of an image by drawing circles or arrows on the captured image.
- Mobile alerts—Unlike a command center based terminal that is constantly monitored, mobile devices will spend most of their time in the pockets of their users, resulting in the user possibly missing critical updates to the COP. As such, users need the ability to notify mobile users of pertinent updates through alerts. To decrease the number of irrelevant notifications, alerts should be specific to users, missions, responder types, or geographical location.
- Links to external programs—Most smartphones are equipped with many useful programs such as calendars, e-mail, text messaging, phone, etc. To further enhance user interface, SPARCCS should include links to these programs in places that they are deemed beneficial.

- Communication with similar systems—As mentioned in Chapter 2, other common operational picture systems exist. Homeland Security is backing the development of the Next Generation Incident Command System. AGIS markets their LifeRing© system that also uses smartphones. The Navy is developing the Command and Control Rapid Prototyping Continuum. While the existence of separate systems is inevitable, there is no reason that these systems shouldn't have the ability to exchange data to enhance the situational awareness provided by each.
- System administrator—Functionality to the web application should be added to allow for a system administrator login. System administrator privileges should include the ability to grant or deny access to applications rather than the autonomous method SPARCCS currently utilizes. Furthermore, system administrators should have the capability to delete missions, points of interest, and images regardless of ownership. And, to ease the process of deleting material, deleting a mission should automatically result in the deletion of all associated points of interest and images.
- Security—Currently, SPARCCS requires a username and password to utilize the application. However, a malicious entity could still externally duplicate HTTP posts and gets. To fix this all HTTP posts and gets should require an autonomous

resubmission of username and password. Another means of security implementation could be a challenge and response method could be implemented in which the response is sent hashed with the user's private key. Furthermore, the current method for storing images is as a separate webpage. While helpful for the application in that this method reduces the amount of data being transferred, it means the images can be viewed from any web browser by entering the image URL. It is important that future developers implement a method to ensure that only registered users have access to images. Finally, SPARCCS needs to implement some form of encryption when transferring data to ensure third parties cannot eavesdrop on sensitive communications.

- HTML 5—The mobile side of SPARCCs was written for the Android operating system due to its user-friendly development process. For SPARCCs to be useful for everyone, the mobile application would need to be rewritten to encompass several makes and models of smartphones. An alternative to this would be to write the application in HTML5. By developing SPARCCS as an HTML5 web application it could be run on any device as long as its browser is HTML5-capable. HTML5 is still in production and is not yet capable of accommodating all of SPARCCS's functionality. However, as it continues

to advance, research should be done to explore the possibility converting SPARCCs to an HTML5 application.

Testing—The only way to truly determine whether the SPARCCS infrastructure meets the requirements needs of users is to conduct field-testing. Mock disaster relief and military scenarios should be performed with responders utilizing SPARCCS. In so doing, key, as well as unneeded or absent, functionality can be identified and built upon. In addition to functionality purposes, battery life testing should also be performed. Key things to consider concerning battery life include sync rates and screen dimming/sleeping times.

The aim of the SPARCCS architecture is to provide a complete Common Operating Picture for those responders working in the field as well as the operators who oversee their missions. Currently implemented in SPARCCS is a solid backbone for a distributed system, utilizing cloud computing in conjunction with mobile devices to enhance situational awareness. The future of the SPARCCS program resides in further implementing current and future mobile and distributed networking technologies to provide a robust platform for users to gain the complete perspective of field operations.

LIST OF REFERENCES

Advanced Ground Information Systems (2010). "AGIS LifeRing Operators Manual." Retrieved from http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&ved=0CEAQjBAwBQ&url=http%3A%2F%2Fwww.agisinc.com%2FAGIS_Operators_manual.pdf&ei=qENNT6S5IYzYiQKQxciaDw&usg=AFQjCNFpgOBznHyUz8SRu_29J62ZsD3iQA&sig2=nvUdf9q5X4lTdqVTK59FuA

Amazon Web Services. "Amazon Simple DB." 2011. Retrived from http://aws.amazon.com/simpledb/

Bain, T. (2009, February 12). *Is the Relational Database Doomed?* Retrieved September 16, 2011, from http://www.readwriteweb.com/enterprise/2009/02/is-the-relational-database-doomed.php

Booze Allen Hamilton. (2010, December 04). *The Economics of Cloud Computing.* Retrieved September 24, 2011, from Addressing the Benefits of Infrastructure in the Cloud: http://www.boozallen.com/media/file/Economics-of-Cloud-Computing.pdf

Comptroller General. (1979 December 14). The World Wide Military Command And Control System—Major Changes Needed In Its Automated Data Processing Management And Direction (LCD-80–22). Washington, DC: Author. Retrieved from http://www.gao.gov/assets/130/128411.pdf.

Copeland, J. (2008, March 25). Emergency Response: Unity of Effort Through A Common Operational Picture. (Master's thesis, U.S. Army War College). Retrieved from http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0CDQQFjAC&url=http%3A%2F%2Fwww.hsdl.org%2F%3Fview%26did=11020&ei=H-RLT5moBpLZiAKH1rm3BA&usg=AFQjCNHnAGmA1zPJ6Bd3wvGmeTqQ4ZMDXg&sig2=xldeA9EyGwIRgTfInVtTKA

Databases. (n.d.). Retrieved September 16, 2011, from Wikipedia: http://en.wikipedia.org/wiki/Database

Fay Chang, J. D. (2006). *Bigtable: A Distributed Storage System for Structured Data.* Retrieved September 16, 2011, from http://static.googleusercontent.com/external_content/untrusted_dlcp/labs.google.com/en/us/papers/bigtable-osdi06.pdf

Department of Defense News Release. (1996 September 26). Global Command and Control System Adopted. Washington, DC: Author. Retrieved from http://www.defense.gov/releases/release.aspx?releaseid=1049

Deputy Secretary of Defense (AT&L). (1971 December 2). World-Wide Military Command and Control System (DoD Directive 5100.30). Washington, DC: Author. Retrieved from http://www.fas.org/spp/military/docops/defense/d5100_30.htm

Duling, J. M. (2009). *THE COMPONENTS NECESSARY FOR SUCCESSFUL INFORMATION SHARING.* (Masters thesis, Naval Post Graduate School). Retrieved from http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCQQFjAA&url=http%3A%2F%2Fwww.dtic.mil%2Fcgi-bin%2FGetTRDoc%3FAD=ADA496711&ei=mYY8T8qPJYmqiQKo7OGlAQ&usg=AFQjCNHqBEKgpn_uQMtkjtLV8JQIbWm0-w&sig2=1yq5Qo8zG7xr8Ny61dC1Bw

Ebbutt, G. (2008). Blue-force tracking evolves for the modern battlefield. *HIS Jane's: Defense & Security Intelligence and Analysis.* Retrieved from http://www.janes.com/products/janes/defence-security-report.aspx?id=1065926194

Endsley, M. R. (1995b). Toward a theory of situation awareness in dynamic systems. *Human Factors, 37,* 32–64.

Google. (2011). *Why App Engine*. Retrieved September 16, 2011, from http://code.google.com/appengine/whyappengine.html

Google. (2011). *Datastore Overview*. Retrieved September 16, 2011, from http://code.google.com/appengine/docs/java/datastore/overview.html

Google. (2011). *Google Maps API Family.* Retrieved September 9, 2011, from http://code.google.com/apis/maps/index.html

Google. (2012). *GWT SDK*. Retrieved January 20, 2012, from http://code.google.com/webtoolkit/learnmore-sdk.html

Google. (2012). *Google Web Toolkit Overview*. Retrieved January 20, 2012, from http://code.google.com/webtoolkit/overview.html

Google. (2012). *Speed Tracer*. Retrieved January 20, 2012, from http://code.google.com/webtoolkit/speedtracer/index.html

Howard, P. (2011, May 17). *When is a Database Not So Relational*. Retrieved September 23, 2011, from http://www.theregister.co.uk/2011/05/17/cloud_databases1/

Lincoln Laboratory. (2011). *Next Generation Incident Command System.* Retrieved September 29, 2011, from http://www.ll.mit.edu/publications/technotes/TechNote_NICS.pdf

Meyer, D. (2011, May 19). *Android shoots past iPhone OS in market share.* Retrieved January 23, 2012, from ZDNet UK / News and Analysis / Mobile IT / Mobile Devices: http://www.zdnet.co.uk/news/mobile-devices/2011/05/19/android-shoots-past-iphone-os-in-market-share-40092829/

Microsoft. (2011). *Cloud Computing.* Retrieved September 24, 2011, from Cloud Basics Government Benfits in the Cloud: http://www.microsoft.com/industry/government/guides/cloud_computing/2-benefits.aspx

Milian, M. (2012, February 03). *CNN .* Retrieved February 03, 2012, from U.S. government, military to get secure Android phones: http://edition.cnn.com/2012/02/03/tech/mobile/government-android-phones/index.html

National Commission on Terrorist Attacks Upon the United States (2004). *The 9/11 Commission Report*. New York: W.W. Norton & Company. Retrieved from http://www.gpoaccess.gov/911/

Tech Target Sites. (2007, December). *Definition Cloud Computing.* Retrieved September 24, 2011, from Search Cloud Computing: http://searchcloudcomputing.techtarget.com/definition/cloud-computing

Tourtelotte, D. R. (2010). *X3D-EARTH: FULL GLOBE COVERAGE UTILIZING MULTIPLE DATASETS.* (Masters thesis, Naval Post Graduate School). Retrieved from: http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCQQFjAA&url=http%3A%2F%2Fedocs.nps.edu%2Fnpspubs%2Fscholarly%2Ftheses%2F2010%2FSep%2F10Sep_Tourtelotte.pdf&ei=VIU8T6HBKOmjiQLpjJWnAQ&usg=AFQjCNHUNmpKNxkGKRMoBQPwFOZcEnpJuw&sig2=Ynf10-Qzw

Vidan, Andy and Gregory Hogan. (2010). “Integrated Sensing and Command and Control System for Disaster Response.” *IEEE*, December 3, 2010. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05655066

W3C Working Group. (2004, February 2004). *W3C Working Group Note 11 February 2004.* Retrieved February 10, 2012, from Web Services Architecture: http://www.w3.org/TR/ws-arch/

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Head, Information Operations and Space Integration
   Branch
   PLI/PP&O/HQMC
   Washington, D.C.